Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA

with Applications for QuantNet 2.0 and GitHub

D I S S E R T A T I O N

zur Erlangung des akademischen Grades

doctor rerum politicarum
(Doktor der Wirtschaftswissenschaft)

eingereicht an der
Wirtschaftswissenschaftlichen Fakultät
der Humboldt-Universität zu Berlin

von
Dipl.-Math. Lukas Borke

Präsidentin der Humboldt-Universität zu Berlin:
Prof. Dr.-Ing. Dr. Sabine Kunst

Dekan der Wirtschaftswissenschaftlichen Fakultät:
Prof. Dr. Christian D. Schade

Gutachter:
1. Prof. Dr. Wolfgang Karl Härdle, Humboldt-Universität zu Berlin
2. Prof. Dr. Stefan Lessmann, Humboldt-Universität zu Berlin

Tag des Kolloquiums: 29. Juni 2017

Abstract

With the growing popularity of GitHub, the largest host of source code and collaboration platform in the world, it has evolved to a Big Data resource offering a variety of Open Source repositories (OSR). At present, there are more than one million organizations on GitHub, among them Google, Facebook, Twitter, Yahoo, CRAN, RStudio, D3, Plotly and many more. GitHub provides an extensive REST API, which enables scientists to retrieve valuable information about the software and research development life cycles. Our research pursues two main objectives: (I) provide an automatic OSR categorization system for data science teams and software developers promoting discoverability, technology transfer and coexistence; (II) establish visual data exploration (VDE) and topic driven navigation of GitHub organizations for collaborative reproducible research (CRR) and web deployment.

To transform Big Data into value, in other words into Smart Data, storing and processing of the data semantics and metadata is essential. Further, the choice of an adequate text mining (TM) model is important. The dynamic calibration of metadata configurations, TM models (VSM, GVSM, LSA), clustering methods and clustering quality indices will be shortened as "smart clusterization". Data-Driven Documents (D3) and Three.js (3D) are JavaScript libraries for producing dynamic, interactive data visualizations, featuring hardware acceleration for rendering complex 2D or 3D computer animations of large data sets. Both techniques enable visual data mining (VDM) in web browsers, and will be abbreviated as D3-3D. Latent Semantic Analysis (LSA) measures semantic information through co-occurrence analysis in the text corpus. Its properties and applicability for Big Data analytics will be demonstrated. "Smart clusterization" combined with the dynamic VDM capabilities of D3-3D will be summarized under the term "Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA".

The introduced "Validation Pipeline" (Vali-PP) is an instrument for "smart clusterization" providing multivariate statistical analysis of the co-occurrence distribution of driving factors of the pipeline. The optimal Vali-PP-configurations of the examined quality indices can be classified into three different categories: GVSM HC, SMART GVSM and MAX HC, which demonstrates that generalized TM models (including LSA), accurate and comprehensive metadata and hierarchical clustering maximize the clustering quality. Three R packages were developed in order to enable GitHub mining and to calibrate the Validation Pipeline, thus establishing an automatic categorization system for OSR – and achieving the primary objective of this thesis. The **rgithubQ** package uses the GitHub API and aims at software mining of GitHub organizations, allowing to implement different parsers for data extraction from various OSR. The **yamldebugger** package facilitates a standardized validation of YAML annotated OSR and acts as a Smart Data extraction layer. The **TManalyzerQ** package constitutes a convenient tool for information retrieval design and performance analysis of different TM models. In this way, the packages **rgithubQ**, **yamldebugger** and **TManalyzerQ** form the basis of a self-contained "GitHub Mining infrastructure in R". Due to the general approach and the underlying R package **tm**, any set of text data can serve as an object of GitHub mining.

The idea of "Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA" is illustrated by visual exploration and topic driven navigation of QuantNet, a GitHub organization consisting of different types of statistics related documents and program codes called Quantlets. Its goal is creating reproducibility and offering a platform for sharing validated knowledge native to the social web. The GitHub API driven QuantNetXploRer and the corresponding D3 based visualization can be found and applied under `http://www.quantlet.de`. The Financial Risk Meter (FRM) is an example for a collaboration project, which was developed over a longer time period by means of the QuantNet platform, thereby allowing CRR. The presented R package **RiskAnalytics** is a convenient tool with the purpose of integrating lasso penalized quantile regression methods with full solution paths and cluster computing support around the topic "Risk Analytics and FRM".

The QuantNet organization serves as a successful reference project for the second objective of this thesis. Every GitHub organization which mimics the YAML Style Guide of QuantNet

can be directly visualized through the QuantNetXploRer.

Keywords: Software Mining, Text Mining, Generalized Vector Space Models, Dimensionality Reduction, YAML, Cluster Analysis, Quality Indices, Validation Pipeline, Cluster and Parallel Computing, Collaborative Reproducible Research, Visual Data Mining, Risk Analytics

Zusammenfassung

Mit der wachsenden Popularität von GitHub, dem größten Online-Anbieter von Programm-Quellcode und der größten Kollaborationsplattform der Welt, hat es sich zu einer Big-Data-Ressource entfaltet, die eine Vielfalt von Open-Source-Repositorien (OSR) anbietet. Gegenwärtig gibt es auf GitHub mehr als eine Million Organisationen, darunter solche wie Google, Facebook, Twitter, Yahoo, CRAN, RStudio, D3, Plotly und viele mehr. GitHub verfügt über eine umfassende REST API, die es Forschern ermöglicht, wertvolle Informationen über die Entwicklungszyklen von Software und Forschung abzurufen. Unsere Arbeit verfolgt zwei Hauptziele: (I) ein automatisches OSR-Kategorisierungssystem für Data Science Teams und Softwareentwickler zu ermöglichen, das Entdeckbarkeit, Technologietransfer und Koexistenz fördert. (II) Visuelle Daten-Exploration (VDE) und thematisch strukturierte Navigation innerhalb von GitHub-Organisationen für reproduzierbare Kooperationsforschung (RKF) und Web-Applikationen zu etablieren.

Um Mehrwert aus Big Data zu generieren, mit anderen Worten Smart Data, ist die Speicherung und Verarbeitung der Datensemantik und Metadaten essenziell. Ferner ist die Wahl eines geeigneten Text Mining (TM) Modells von Bedeutung. Die dynamische Kalibrierung der Metadaten-Konfigurationen, TM Modelle (VSM, GVSM, LSA), Clustering-Methoden und Clustering-Qualitätsindizes wird als "Smart Clusterization" abgekürzt. Data-Driven Documents (D3) und Three.js (3D) sind JavaScript-Bibliotheken, um dynamische, interaktive Datenvisualisierung zu erzeugen, die sich durch Hardwarebeschleunigung der komplexen 2D- und 3D-Computeranimationen von großen Datenmengen auszeichnet. Beide Techniken erlauben Visuelles Data Mining (VDM) in Webbrowsern, und werden als D3-3D abgekürzt. Latent Semantic Analysis (LSA) misst semantische Information durch Kontingenzanalyse des Textkorpus. Ihre Eigenschaften und Anwendbarkeit für Big-Data-Analytik werden demonstriert. "Smart clusterization", kombiniert mit den dynamischen VDM-Möglichkeiten von D3-3D, wird unter dem Begriff "Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA" zusammengefasst.

Die eingeführte "Validation Pipeline" (Vali-PP) ist ein Instrument für "Smart Clusterization", das multivariate statistische Analyse der gemeinsamen Verteilung der treibenden Faktoren der Pipeline erlaubt. Die optimalen Vali-PP-Konfigurationen der untersuchten Qualitätsindizes können in drei unterschiedliche Kategorien eingeordnet werden: GVSM HC, SMART GVSM und MAX HC, was verdeutlicht, dass verallgemeinerte TM Modelle (einschleßlich LSA), präzise und umfassende Metadaten und hierarchische Clustering-Verfahren die Clustering-Qualität maximieren. Drei R-Pakete wurden entwickelt, um GitHub Mining zu ermöglichen und die Validation Pipeline zu kalibrieren, wodurch ein automatisches Kategorisierungssystem für OSR geschaffen – und das primäre Hauptziel dieser Arbeit erreicht wird. Das **rgithubQ** Paket benutzt die GitHub API und zielt auf Software Mining der GitHub-Organisationen ab, dabei lässt es die Implementierung verschiedener Parser für Datenextraktion aus diversen OSR zu. Das **yamldebugger** Paket erleichtert eine standardisierte Validierung von OSR, die mit YAML ausgezeichnet sind, und fungiert als ein Smart-Data-Extraktionslayer. Das **TManalyzerQ** Paket stellt ein geeignetes Werkzeug für den Entwurf von Information Retrieval-Systemen und die Performanceanalyse unterschiedlicher TM Modelle dar. Auf diese Weise bilden die Pakete **rgithubQ**, **yamldebugger** und **TManalyzerQ** die Grundlage einer eigenständigen "GitHub Mining Infrastruktur in R". Wegen des allgemeinen Ansatzes und des zugrunde liegenden R-Pakets **tm** können beliebige Textdaten als Studienobjekt des GitHub Minings dienen.

Die Idee des "Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA" wird veranschaulicht durch visuelle Exploration und thematisch strukturierte Navigation von QuantNet, einer GitHub-Organisation, bestehend aus verschiedenen Arten von statistikbezogenen Dokumenten und Programmcodes, die Quantlets genannt werden. Ihr Ziel ist es, Reproduzierbarkeit zu schaffen und eine Plattform anzubieten, die validiertes und im Social Web beheimatetes Wissen teilt. Der durch die GitHub API angetriebene QuantNetXploRer und die entsprechene D3-basierte Visualisierung können unter `http://www.quantlet.de` gefunden und angewendet werden. Der Financial Risk Meter (FRM) ist ein Beispiel für ein

Kooperationsprojekt, das über eine längere Zeit mit Hilfe der QuantNet-Plattform entwickelt wurde, wodurch RKF praktiziert wird. Das vorgestellte R-Paket **RiskAnalytics** ist ein geeignetes Hilfsmittel mit dem Zweck, Methoden zur Quantil-Regression mit LASSO-Regularisierung und vollem Lösungsverlauf sowie Cluster-Computing-Unterstützung rund um das Thema "Risk Analytics and FRM" zu integrieren.

Die QuantNet-Organisation dient als ein erfolgreiches Referenzprojekt für das zweite Hauptziel dieser Arbeit. Jede GitHub-Organisation, die den YAML-Styleguide von QuantNet nachahmt, kann direkt durch den QuantNetXploRer visualisiert werden.

Schlagwörter: Software Mining, Text Mining, Verallgemeinerte Vektorraum-Modelle, Dimensionsreduktion, YAML, Clusteranalyse, Qualitätsindizes, Validierungs-Pipeline, Verteilt-paralleles Rechnen, Reproduzierbare Kooperationsforschung, Visuelles Data Mining, Risk Analytics

Acknowledgement

I would like to thank to my Ph.D. advisor, Professor Härdle[1], for the hard question which incented me to widen my research from various perspectives. He is someone who stands out from the crowd and whom you'll never forget once you meet him.

I am thankful to my colleagues from the Ladislaus von Bortkiewicz Chair of Statistics[2] and from the Research Data Center[3] for guiding the focus of my research so that I could address some real-world problems.

I am particularly grateful to Svetlana Bykovskaya[4] for her assistance in developing a new QuantNet infrastructure on GitHub, updating the data from QuantNet according to the new format and migrating them into GitHub. Her participation in the programming of the D3 visualization and her contribution to the general idea of the "GitHub Mining Infrastructure in R" is equally appreciated.

Next, I would like to thank the entire CRAN community[5] for the various and highly useful R packages which have promoted and enriched my developing progress of the "GitHub Mining" idea, and thus the overall success of my research.

Financial support from the Deutsche Forschungsgemeinschaft via CRC "Economic Risk"[6] and IRTG 1792 "High Dimensional Non Stationary Time Series"[7], Humboldt-Universität zu Berlin, is gratefully acknowledged.

Finally, I thank my family for their support during the last years. In particular, my daughter Sophie helped me to set the right priorities.

[1] http://hu.berlin/93629
[2] https://www.wiwi.hu-berlin.de/de/professuren/quantitativ/statistik
[3] http://sfb649.wiwi.hu-berlin.de/fedc/team.php
[4] https://github.com/polarstern
[5] https://cran.r-project.org/
[6] http://sfb649.wiwi.hu-berlin.de/index.php
[7] https://www.wiwi.hu-berlin.de/de/forschung/irtg

Contents

List of abbreviations

Clustering related

$\mathcal{C}$	The result of a clustering (also called clustering), a set of clusters C
HC	Hierarchical clustering
M^3	M^3 evaluation: TM models, clustering methods and validation measures are combined in a $3 \times 3 \times 3$ benchmark setup
$M^4_{d_1,d_2,d_3,d_4,max}$	Validation Pipeline benchmark dealing with the following 5 dimensions: metadata configurations, TM models, clustering methods, clustering quality indices and number of clusters
PAM	Partitioning around medoids
Vali-PP	Validation Pipeline

IT related

API	Application programming interface
IDE	Integrated development environment
JSON	JavaScript object notation: an open-standard format that uses human-readable text to transmit data objects consisting of attribute-value pairs
VC	Version control
YAML	YAML Ain't Markup Language: a human-readable data serialization language

QuantNet related

Q3	GitHub API based QuantNet Mining infrastructure in R
Q3-D3-LSA	Combination of "GitHub API based QuantNet Mining infrastructure in R", D3 implementation and LSA
QL	Quantlets: empirical as well as quantitative-theoretical methods for statistical and economical programming

Reproducible Research related

CRAN	Comprehensive R Archive Network
CRR	Collaborative reproducible research
GiHuMiR	GitHub Mining infrastructure in R
OSR	Open source repositories

Risk Analytics related

FRM	Financial Risk Meter

NASDAQ is the acronym of National Association of Securities Dealers Automated Quotations; the Nasdaq Stock Market is an American stock exchange

QR Quantile regression

S&P 500 The Standard & Poor's 500, or just "the S&P", is an American stock market index based on the market capitalizations of 500 large companies having common stock listed on the NYSE or NASDAQ

VIX is the ticker symbol for the CBOE Volatility Index, a popular measure of the implied volatility of S&P 500 index options

Text Mining related

BVSM Basic vector space model

D Term document matrix (as mathematical object)

GVSM Generalized vector space model

GVSM(TT) Term-term correlations GVSM

idf Inverse document frequency

IR Information retrieval

LSA Latent Semantic Analysis

M_D Distance matrix

M_S Similarity matrix

PC Semantic space Principal Components

SVD Singular value decomposition

T Set of terms/vocabulary

TDM Term document matrix

TM Text mining

tf Absolute term frequency

VSM Vector space model

Visual Data Mining related

C3 D3-based reusable chart library

D3 D3.js (or just D3 for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers

D3-3D JavaScript libraries D3.js and Three.js

D3-3D-LSA Technology comprising the following main components: D3.js, Three.js and LSA

MDS	Multidimensional scaling
SVG	Scalable vector graphics
t-SNE	t-Distributed Stochastic Neighbor Embedding
VDE	Visual data exploration
VDM	Visual data mining
WebGL	Web Graphics Library is a JavaScript API for rendering 3D graphics within any compatible web browser without the use of plug-ins

List of Figures

List of Tables

Listings

1 Introduction

1.1 Collaborative Reproducible Research (CRR)

> Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.
>
> *Literate programming (1984)*
> Donald Knuth

For collaboration, researchers tend to use a mix of e-mail, version control (VC) systems and shared network folders (Dropbox, etc.). VC systems are critically important in making research collaborative and reproducible. They allow groups to work collaboratively on documents and track how they evolve over time. Ideally, all aspects of computational research would be hosted on publicly available VC repositories, such as GitHub or BitBucket.

In accordance with good research practices reproducibility is one main aspect for software and methodological platforms on which reproducible research can be conducted and distributed. The entire life cycle of scientific research can be summarized in 5 steps as following: individual exploration, collaboration, production-scale execution, publication, education (Stodden et al., 2014).

Reproducibility is the ultimate standard by which scientific findings are judged. From the computer science perspective, reproducible research is often related to literate programming[1], a paradigm conceived by Donald Knuth. The basic idea is to combine computer code and software documentation in the same document.

In recent years, a number of web services have appeared that play the role of a central hub of distributed VC. GitHub is the most popular of them, but others such as BitBucket and GitLab are possible alternatives. GitHub has had a tremendous impact in the open-source community, reaching in a few years millions of active users and gaining rapidly popularity in scientific circles. The core feature of the collaborative process on GitHub is known as a pull request, which is comparable to a public peer review of a set of changes to a manuscript.

1.2 Smart Data

Big Data The result of an extensive literature review on Big Data definitions by De Mauro et al. (2015) concluded that a consensual definition of Big Data would be that "Big Data represents the Information assets characterized by such a High Volume, Velocity and Variety to require specific Technology and Analytical Methods for its transformation into Value". According to Hilbert (2016), the quantitative explosion of three kinds of digital capacities, namely Information flow, Information stock and Information computation, has led to five distinguished qualitative characteristics in the way data is treated:

1. Big Data is produced anyways
2. Big Data replaces random sampling
3. Big Data is often accessible in real-time
4. Big Data merges different sources

[1] `http://www.literateprogramming.com/`

5. The full name of Big Data is Big Data Analytics.

The current Wikipedia definition[2] describes Big Data by the following characteristics:

- Volume: The quantity of generated and stored data.
- Variety: The type and nature of the data.
- Velocity: The speed at which the data is generated and processed.
- Variability: Inconsistency of the data set.
- Veracity: The quality of captured data can vary greatly, affecting accurate analysis.

From Big Data to Smart Data Bernard Bekavac[3] proposed the following intuitive definition of Smart Data:

- Data is self-explanatory
- Data is given meaning (by attaching semantic description of its content)
- The presence of formal description of the semantics (OWL, RDF etc.) enables automated processing of data
- (Open) Data is interconnected by means of URI/HTTP

The Smart Data experts from trommsdorff + drüner[4] argue that the common goal must be to move from the established input-orientated definition of Big Data, through the so called "3V-perspective" (volume, velocity and variety of data), to a more output-oriented perspective, where the concrete questions to be answered are defined before the data is gathered. Through this a fourth V: Value can be developed.

According to trommsdorff + drüner, the new 4P's of data-driven marketing are Purpose, People, Process, Platform. Only when is defined for what Purpose data is processed, People can define Processes, thereby making best use of the Smart Data. Then they can establish the appropriate IT equipment (Platform), thus achieving an optimum yield of Smart Data across all processing steps.

From Smart Data to Metadata The Smart Data Memorandum[5], which is an initiative of the Trusted Cloud[6] research, discusses the specifications and definitions of the term "Smart Data" regarding the delimitation of the term "Big Data". The initiators of the Memorandum establish the concise formula:

Smart Data
= Big Data + Value + Semantics + Data quality + Security + Data protection
= useful, high-quality and secured/verified Data

The German original version reads as follows:

Smart Data
= Big Data + Nutzen + Semantik + Datenqualität + Sicherheit + Datenschutz
= nutzbringende, hochwertige und abgesicherte Daten

Commonly, data sets and stocks are already available or are easily obtained, and possess already some usable structure. Added value of Smart Data can be only gained by

[2] https://en.wikipedia.org/wiki/Big_data
[3] http://www.sbt.ti.ch/doc/forum/Herbstschule-2012/Symposium/Big_Smart_Data_2012_druck.pdf
[4] https://www.td-berlin.com/
[5] http://smart-data.fzi.de/
[6] https://www.trusted-cloud.de/

- storing and processing of the data semantics and Metadata during the data processing cycle, as well as
- defined quality characteristics of data

Metadata is "data [information] that provides information about other data"[7]. It describes the data quality, data structure, semantics, origin of the data, purpose, usage rights or also data protection obligations. Therefore, Metadata is essential for the intelligent evaluation and correct utilization of the base data. Metadata is the "Smartness" in Smart Data. However, the form and detail degree of the Metadata depends strongly on the intended use (Purpose) of the base data.

Section 3.4 discusses YAML, which is a human friendly data serialization standard. Designed as a human-readable and data-oriented language in 2001, YAML can easily be applied to widely used data frames such as lists and arrays and matches the native data structures of agile languages. Additionally, there exist many YAML parser implementations for various programming languages, accompanied by the fact that YAML data is portable between them. Furthermore, there are already numerous GitHub organizations using YAML for Metadata. Due to the properties mentioned above, YAML was selected as annotation language for OSR Metadata in our research. Section 3.4.3 introduces the **yamldebugger** package, which was written in order to facilitate a standardized validation and analysis of YAML annotated OSR.

1.3 D3-3D-LSA

Keim (2002) worked out the following main advantages of visual data exploration (VDE) over automatic data mining techniques. First, VDE can easily handle highly inhomogeneous and noisy data. Second, VDE is intuitive and requires no understanding of complex techniques and theories from domains like mathematics, statistics, information and computer science.

The "Dynamic Clustering and Visualization" of the examined Smart Data examples is performed by means of the "D3-3D-LSA" technology. D3-3D-LSA comprises the following main components:

- D3: D3.js[8] – Data-Driven Documents
  - VDE via information visualization in two dimensions
- 3D: Three.js[9] – visual data mining and exploration in 3D
  - cross-browser JavaScript library/API using the WebGL[10] standard
  - animates 3D computer graphics in a web browser
- LSA[11]: Latent Semantic Analysis, a TM model measuring semantic information through co-occurrence analysis in the text corpus

D3 D3 is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It provides a huge collection of visualization examples: from simple charts and diagrams to highly sophisticated real-world applications like networks, force-directed graph clusters, maps and other geographic visualizations[12]. Various applications which are powered by D3 encompass: C3.js (`http://c3js.org/`) - a reusable chart library; GitHub Visualizer[13] which generates and animates statistics of repositories' history and collaboration timelines on

[7] `https://en.wikipedia.org/wiki/Metadata`
[8] `https://github.com/d3`
[9] `https://github.com/mrdoob/three.js`
[10] `https://www.khronos.org/webgl/`
[11] `https://github.com/cran/lsa`
[12] `https://github.com/d3/d3/wiki/Gallery`
[13] `http://ghv.artzub.com/`

GitHub; RCloud[14] (an Integrated Exploratory Analysis, Visualization, and Deployment on the Web) providing an environment in which R packages can create rich HTML content; Plotly (`https://plot.ly/`), an online analytics and data visualization tool providing statistics tools for individuals and collaboration, as well as scientific graphing libraries for Python, R, MATLAB, Perl, Julia, Arduino, and REST (`https://api.plot.ly/v2/`). Section 2.3 contains detailed information about the development of the main D3 components for the QuantNet visualization together with live examples on GitHub pages. Additionally, the QuantNetXploRer is a good example of D3 in power, see Section 2.7.

3D Three.js enables interactive visualization of high-dimensional data representing the similarity (as well as distance) of data points. Its graphics engine is realized via a cross-browser JavaScript library, which provides hardware acceleration for rendering complex 3D computer animations of very large data sets. A comprehensive collection of projects powered by Three.js may be found on their website (`https://threejs.org/`). Examples of these include, among many others:

- Data Projector[15] visualizing High-Dimensional Data through SVD and t-SNE
- **threejs**[16], an R package providing interactive 3D scatterplots and globe plots
- Software Galaxies[17] combining 3D visualizations of major software package managers:
  Python (`https://www.python.org/`),
  CRAN package network (`https://cran.r-project.org/`) of R packages,
  npm (`https://www.npmjs.com/`), and others
- GitHub followers visualization[18] showing all GitHub users who have more than two followers, was thankfully adopted from `https://github.com/anvaka/allgithub`
- The repository Plotly containing some D3 and 3D Plotly examples.

LSA To increase the information retrieval (IR) efficiency there is a need for incorporating semantic information. In Section 2.4.4, three text mining (TM) models will be examined: vector space model (VSM), generalized VSM (GVSM) and latent semantic analysis (LSA). The LSA has been successfully used for IR purposes as a technique for capturing semantic relations between terms and inserting them into the similarity measure between documents. The main advantage of LSA is the flexible dimension reduction property, which is controlled by the truncation parameter k within the SVD process, and its applicability for Big Data. Additionally, the M^3 evaluation identifies the LSA/LSA50 and hierarchical clustering (HC) as the optimal model/method combination, see Section 2.6.3.

Our cluster validation results show that different TM model configurations allow adapted similarity-based document clustering and knowledge discovery. Further, we introduce the ***Validation Pipeline*** (*Vali-PP*) in Section 3.7 and apply it on the YAML Smart Data. *Vali-PP* is a functional multi-staged instrument for clustering analysis, providing multivariate statistical analysis of the co-occurrence distribution of driving factors of the pipeline. The optimal *Vali-PP*-configurations of the examined quality indices can be classified into three different categories: GVSM HC, SMART GVSM and MAX HC, which demonstrates that generalized TM models (GVSM including LSA), accurate and comprehensive metadata (SMART, MAX) and hierarchical clustering maximize the clustering quality, see Section 3.7.5.

The k dimensions of the LSA space can be interpreted as the main semantic components. The "LSA anatomy" and the "semantic kernel" are examined in depth in Section 2.6.4. For

[14] `https://github.com/att/rcloud`
[15] `https://github.com/datacratic/data-projector`
[16] `http://bwlewis.github.io/rthreejs/`
[17] `https://github.com/anvaka/pm`
[18] `http://borke.net/PackageNetwork/#/galaxy/github`

the purpose of interactive and dynamic VDM, the LSA space will be further projected on two and three dimensions. Applying projection techniques like multidimensional scaling (MDS) or t-distributed stochastic neighbour embedding (t-SNE) (see Section 2.5), optimal clustering configurations (see Section 3.7) and the aforementioned D3-3D visualization components allows to implement various interactive applications encompassing IR and document clustering functionality.

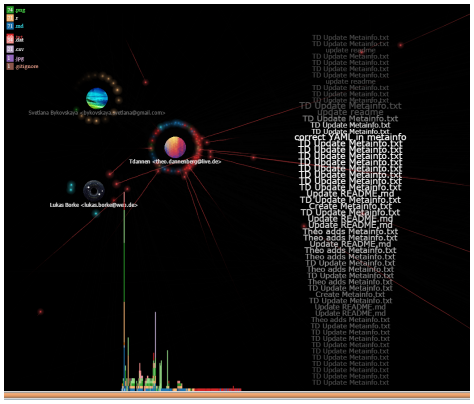


Figure 1.1: Collaboration Timeline of the MVA repository via GitHub Visualizer

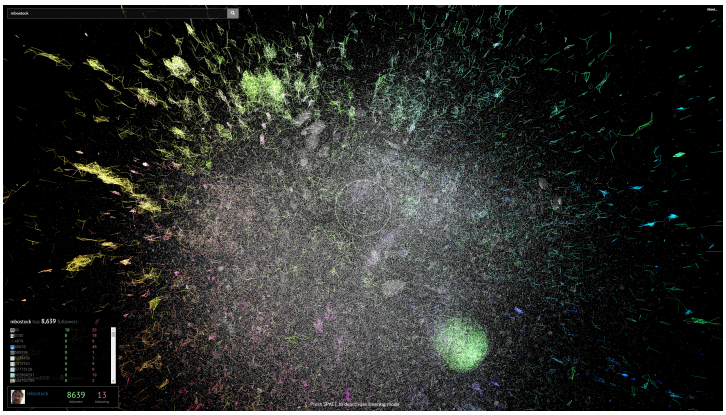


Figure 1.3: 3D GitHub followers visualization showing Mike Bostock

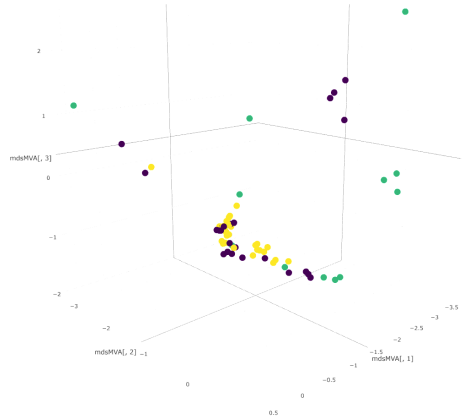


Figure 1.2: k-means clustering and metric MDS for MVA quantlets via Plotly

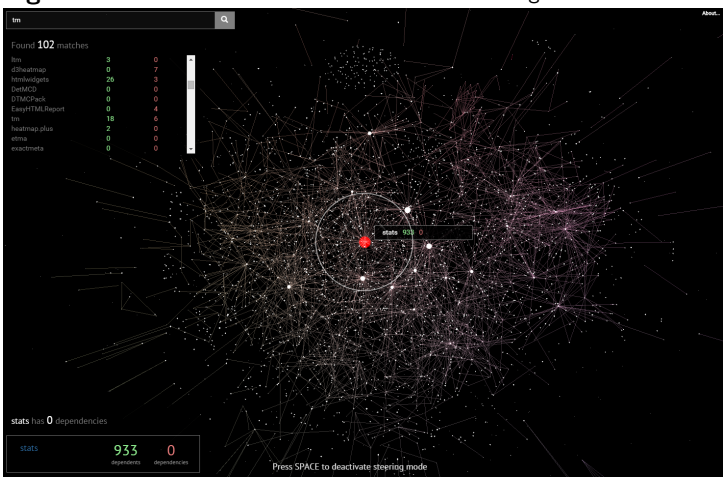


Figure 1.4: 3D CRAN Network Graph - R Language

Figures 1.1 and 1.3 are available as animation in Git2Q3-Collaboration, the visualization from Figure 1.2 is available as interactive MVAQnetClusKmeans_plotly. The dynamic 3D graphs from Figures 1.3 and 1.4 can be accessed at `http://borke.net/PackageNetwork/`, they were thankfully adopted from Andrei Kashcha[19].

> The **dynamic** calibration of metadata configurations, vector space models (VSM, GVSM, **LSA**), **clustering methods**, **clustering quality indices** and number of **clusters** resulting in a **smart clusterization**, together with the **dynamic visual data mining** capabilities of **D3-3D**

this is what we call:
"Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA".

[19] `https://github.com/anvaka`

1.4 QuantNet, GitHub, applications

QuantNet being an online GitHub organization is an integrated environment consisting of different types of statistics-related documents and program codes called Quantlets. Its goal is creating reproducibility and offering a platform for sharing validated knowledge native to the social web. QuantNet and the corresponding D3 based visualization can be found and applied under `http://quantlet.de`. The driving technology behind it is Q3-D3-LSA, which is the combination of "GitHub API based QuantNet Mining infrastructure in R" (covered in Chapter 3), D3 implementation and LSA.

The new package **rgithubQ**, which enables a GitHub wide search for code and repositories using the GitHub Search API and which is an essential element of the QuantNet Mining infrastructure, is briefly presented in Section 3.3. The QuantNet Style Guide and the **yamldebugger** package allow a standardized audit and validation of YAML annotated OSR, see Section 3.4. The behavior statistics of QuantNet users are measured with Web Metrics from **Google Analytics** in Section 3.5. We show in Section 3.6 how the search queries obtained from Google's metrics can be used in the *test collections* in order to calibrate and evaluate the information retrieval (IR) performance of QuantNet's search engine called QuantNetXploRer. For that purpose, different TM models will be examined by means of the new **TManalyzerQ** package.

The **TManalyzerQ** results show that for all considered single term queries the number of true positives is maximal in a latent semantic analysis model configuration (LSA50). Subsequently, the findings of our experimental design are implemented into the QuantNetXploRer. The GitHub API driven QuantNetXploRer can be found and mined under `http://quantlet.de`.

The Financial Risk Meter (FRM) project (Yu et al., 2017) is an example for a collaboration project, which was developed over a longer time period by means of the QuantNet platform, thereby allowing CRR. In order to integrate and facilitate the research, calculation and analysis methods around the FRM project, the R package **RiskAnalytics** has been developed (Borke, 2017a). This package is presented in Chapter 4. Its main goal is to provide data processing and parallelized quantile lasso regression methods for risk analysis based on NASDAQ data, Yahoo Finance data and some macro variables. The derived "Risk Analytics" in Section 4.3.4 can help to forecast and evaluate the systemic risk for the corresponding markets. The D3 based visualization and the up-to-date FRM can be found on `http://frm.wiwi.hu-berlin.de`. Supplementary R codes are published on www.quantlet.de with the keyword QFRM. The **RiskAnalytics** package is a convenient tool with the purpose of integrating lasso penalized quantile regression methods with full solution paths and cluster computing support around the topic "Risk Analytics and FRM". Additionally, the interactive and web based ***RiskAnalytics*** *scientific IDE* combines the scientific, technical and visual materials, elements and sources around the research field "Risk Analytics", see Section 4.4. Figure 1.5 demonstrates the combined utilisation of the concept of "Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA" and the QuantNetXploRer, showing how desired statistical methods can be mined, clustered and visualized.

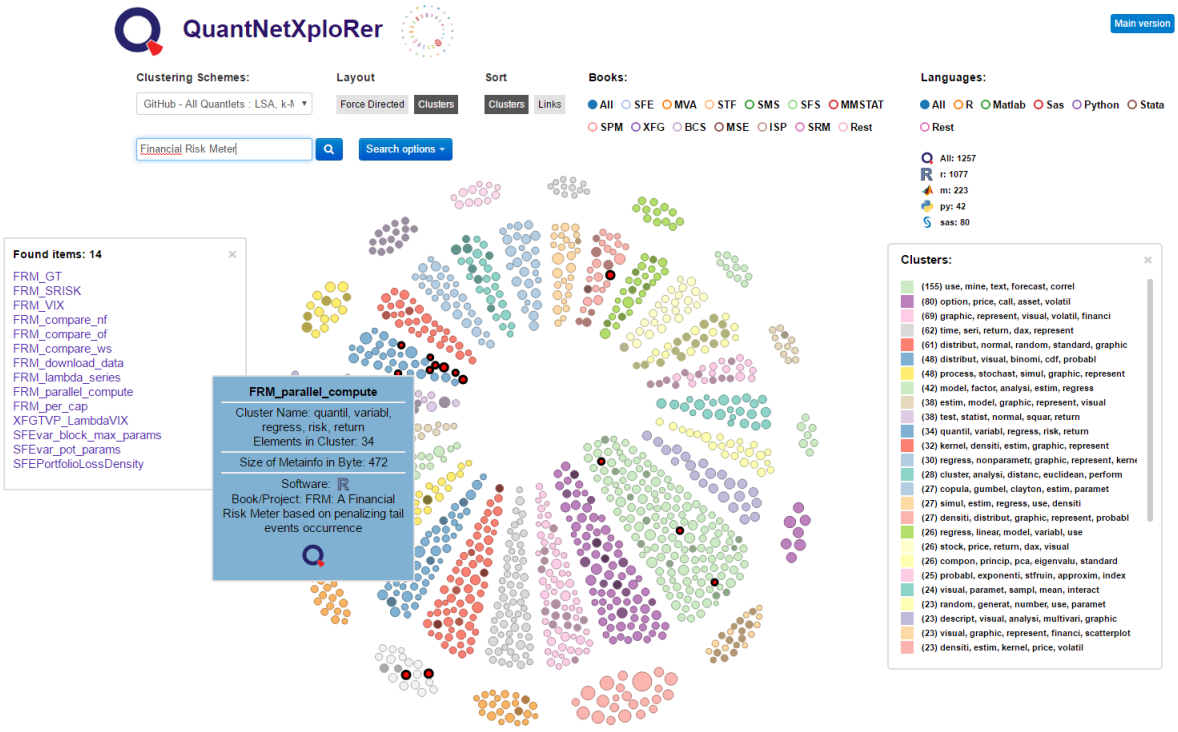


Figure 1.5: All Quantlets in the current QuantNetXploRer D3 visualization, search term "Financial Risk Meter", the most FRM related Quantlets are concentrated in the cluster "quantil, variabl, regress, risk, return"

2 Q3-D3-LSA

2.1 Introduction – From Data to Information

The "QuantNet" concept is the effort to collect, interlink, retrieve and visualize all the information in the scientific community with the particular emphasis on statistics. The richness and diversity of various and heterogeneous data types, descriptions and data sets submitted by numerous authors require an appropriate text mining model to be established and tuned. The big collection of data has now to be distilled to human-readable and applicable information and at the same time a modern and robust visualization framework is crucial.

QuantNet was originally designed as a platform to freely exchange empirical as well as quantitative-theoretical methods, called Quantlets. It supported the deployment of computer codes (R, Matlab, SAS and Python), thus helping to establish collaborative reproducible research (CRR) in the field of applied statistics and econometrics at the Collaborative Research Center 649[1], operated at the Humboldt University of Berlin. The former PHP-based QuantNet provided users a series of basic functions including registration, Quantlet uploading, searching, demonstrating and downloading. Heterogeneous resources submitted by diverse contributors were stored on a proprietary Linux server having its own Oracle database. Hence, this IT-infrastructure was quite restrictive, maintenance-intensive and also relatively susceptible to errors due to strict data type requirements, complexity and constraints of the Oracle database.

With the time, some problems and drawbacks became increasingly apparent:

1. lack of version control (VC) and source code management (SCM)
2. lack of distinct abilities of collaboration and project management between teams and heterogeneous groups of people
3. high personal maintenance costs of the infrastructure
4. database-restrictions and inflexibility of data handling
5. lack of a clear abstraction barrier between the data storage and the text mining (TM) and visualization layer of the system architecture

The points 1, 2 and 3 could be easily solved by the immanent features of the "GitHub's philosophy". As Marcio von Muhlen[2] (Product Manager at Dropbox) eloquently expresses:

> GitHub is a social network of code, the first platform for sharing validated knowledge native to the social web. Open Science efforts like arXiv and PLoS ONE should follow GitHub's lead and embrace the social web.

Point 4 could be tackled by using the YAML standard (`http://yaml.org/`) for meta information of the resources, thus replacing the necessity of a database system. More about this human-readable data serialization language can be found on GitHub[3]. Point 5 could be realized via the GitHub API (Cosentino et al., 2016). After the challenge of the abstraction barrier was solved, it was a straightforward procedure to connect the newly created Quantlet organization[4] on GitHub with the rest of the existing system architecture, comprising the TM and D3.js visualization layer.

[1] `http://sfb649.wiwi.hu-berlin.de/`
[2] `http://marciovm.com/`
[3] `https://github.com/yaml/yaml-spec`
[4] `https://github.com/Quantlet`

QuantNet[5] is now an online GitHub based organization with diverse repositories of scientific information consisting of statistics related documents and program codes. The advantages of QuantNet are:

- Full integration with GitHub
- Proprietary GitHub-R-API implementation developed from the core R package **github** (Scheidegger, 2016) available as GitHub repository "R Bindings for the Github v3 API" (`https://github.com/cscheid/rgithub`) from Carlos Scheidegger, professor in the Department of Computer Science at the University of Arizona
- TM Pipeline providing IR, document clustering and D3 visualizations realized via QuantMining, a "GitHub API based QuantNet Mining infrastructure in R"
- Tuned and integrated search engine within the main D3 Visu based on validated meta information in Quantlets
- Ease of discovery and use of your technology and research results, everything in a single GitHub Markdown page
- Standardized audit and validation of your technology by means of the Style Guide[6] and Yamldebugger[7] (Borke, 2017d)

2.1.1 Transparency, Collaboration and Reproducibility

QuantNet – open access code-sharing platform:

- Quantlets: R, Matlab, SAS and Python programs, various authors and topics
- QuantNetXploRer: Q3-D3-LSA driven and GitHub based search engine
- Knowledge discovery of brand-new research topics but also of dormant and archived research materials as required by good scientific practice

The Q3-D3-LSA technology comprises the following main components:

- Q3 (Quantlets, QuantNet, QuantMining): Scientific data pool and data mining infrastructure for CRR
- D3 (Data-Driven Documents): Knowledge discovery via information visualization by use of the D3 JavaScript library combining powerful visualization components and a data-driven approach
- LSA (Latent Semantic Analysis): Semantic embedding for higher clustering performance and automatic document classification by topic labeling

2.2 Related Work

Feinerer and Wild (2007) applied LSA based algorithms in a fully automated way on transcripts

[5] `http://quantlet.de`
[6] `https://github.com/Quantlet/Styleguide-and-FAQ`
[7] `https://github.com/Quantlet/yamldebugger`

of interviews. The machine results were compared against marketing expert judgments with the outcome that the proposed algorithms provided perfect reliability with appropriate validity in automated coding and textual analysis. Feinerer and Wild (2007) could guarantee reliability on a very high level, while at the same time avoiding the main disadvantages of qualitative methods performed by humans like their inherent subjectivity and their high costs.

Linstead et al. (2008) pointed out that while there has been progress in developing sourcecode-specific search engines in recent years (e.g. Koders, Krugle, and Google's CodeSearch), these systems continue to focus strictly on text information retrieval, and do not appear to leverage the copious relations that can be extracted and analyzed from code. By combining software textual content with structural information captured by their CodeRank approach, they were able to significantly improve software retrieval performance. Developing and applying probabilistic models to automatically discover the topics embedded in the code and extracting topic-word and author-topic distributions, the authors provided a statistical and information-theoretic basis for quantifying and analyzing developer similarity and competence, topic scattering, and document tangling, with direct applications to software engineering.

Encouraged by the presented studies, we propose in this paper to use the latent semantic analysis (LSA) (Deerwester et al., 1990) as a technique capturing semantic relations between terms and inserting them into the similarity measure between documents. In this approach, the documents are implicitly mapped into a "semantic space", where documents that do not share any terms can still be close to each other if their terms are semantically related. The semantic similarity between two terms is inferred by an analysis of their co-occurrence patterns: terms that co-occur often in the same documents are considered as related. This statistical co-occurrence information is extracted by means of a singular value decomposition (SVD) of the "term by document" matrix in the way described in Section 2.4.

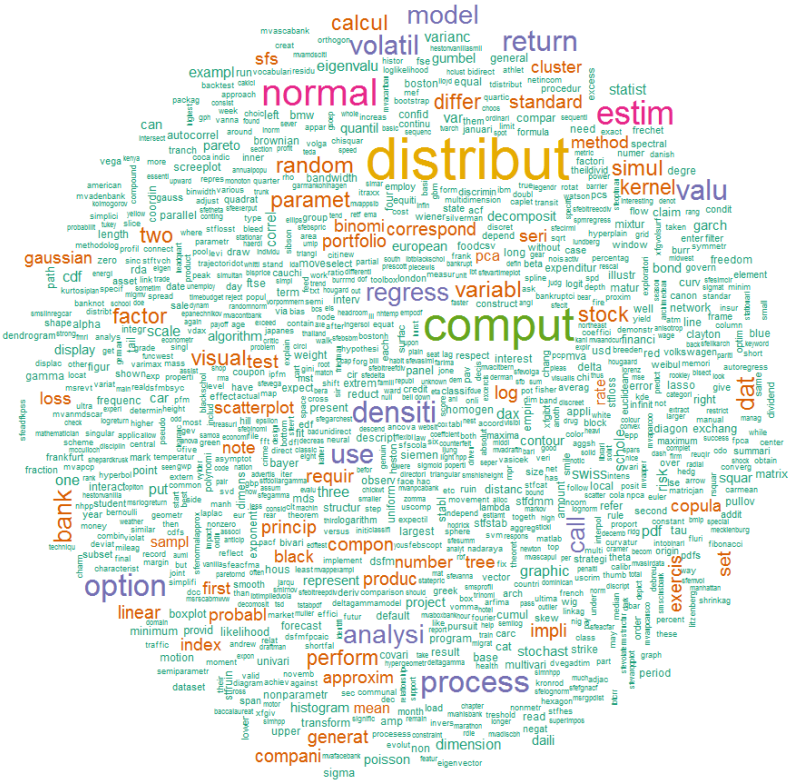


Figure 2.1: Word cloud of the QuantNet terms

2.3 Q3-D3 Genesis

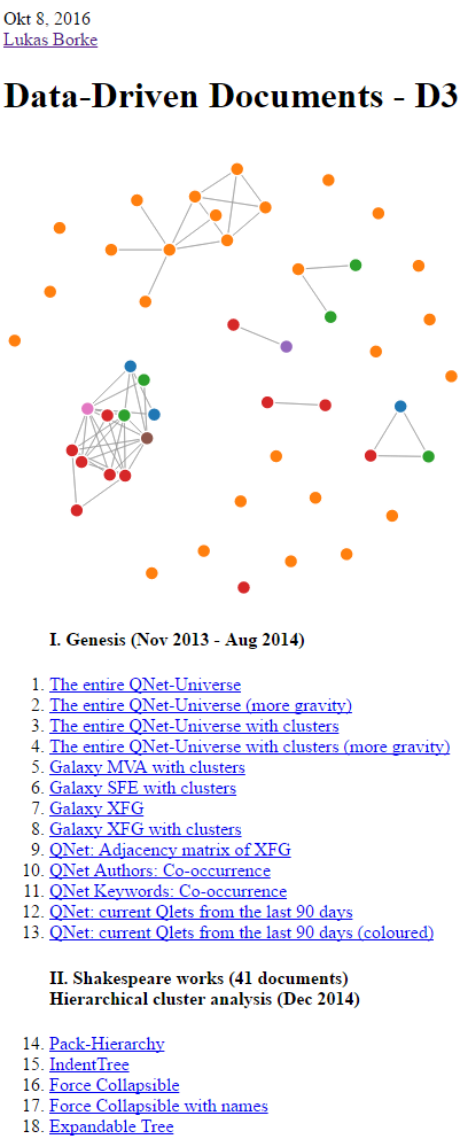


Figure 2.2: Q3-D3 Genesis - Chapters

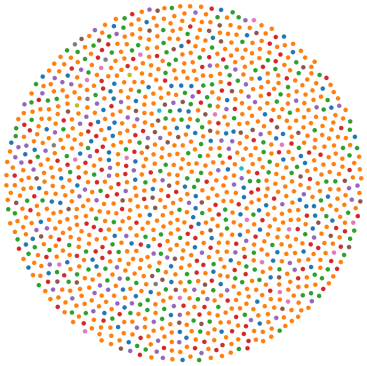


Figure 2.3: The entire QNet-Universe

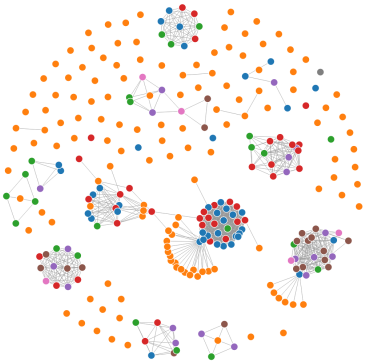


Figure 2.4: Galaxy MVA with clusters

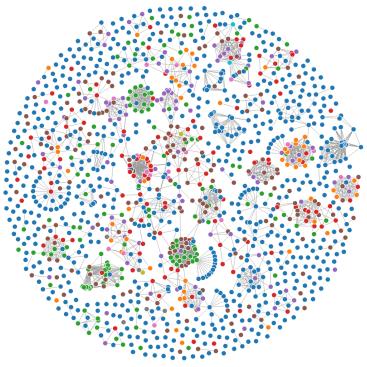


Figure 2.5: The entire QNet-Universe with clusters

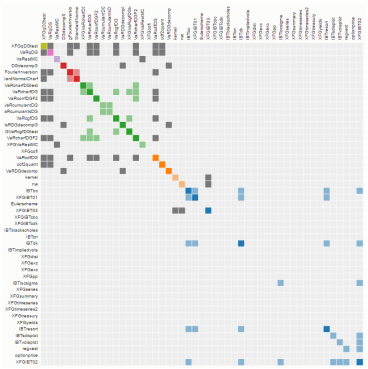


Figure 2.6: Adjacency matrix of XFG

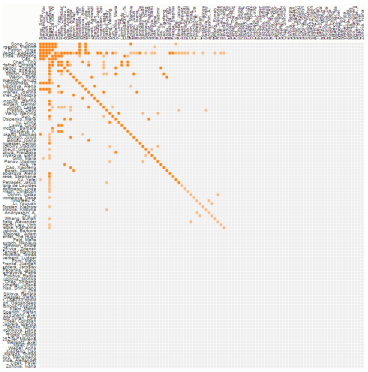


Figure 2.7: Authors: Co-occurrence

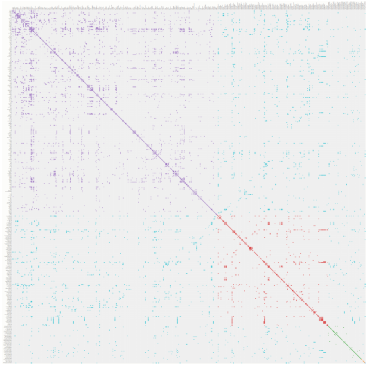


Figure 2.8: Keywords: Co-occurrence

D3 (`https://d3js.org/`) is a rather new and not traditional visualization framework introduced by Bostock et al. (2011). D3.js (or D3 for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. It makes use of the widely implemented SVG, HTML5, and CSS standards. Instead of establishing a novel graphical grammar, D3 solves a different, smaller problem: efficient manipulation of documents based on data. The software design is heavily influenced by prior visualization systems, including Protovis.

The D3 gallery (available at `http://bl.ocks.org/mbostock`) demonstrates diverse capabilities and performance of the D3 technology, providing a huge collection of D3 visualization

examples. Moreover, various applications and frameworks for data visualization have been built using D3, combining its methods with other modern technologies. Examples of these include, among many others, a data visualization library Plotly (see `https://plot.ly`) and a Force-directed Network Visualization developed by Jim Vallandingham[8].

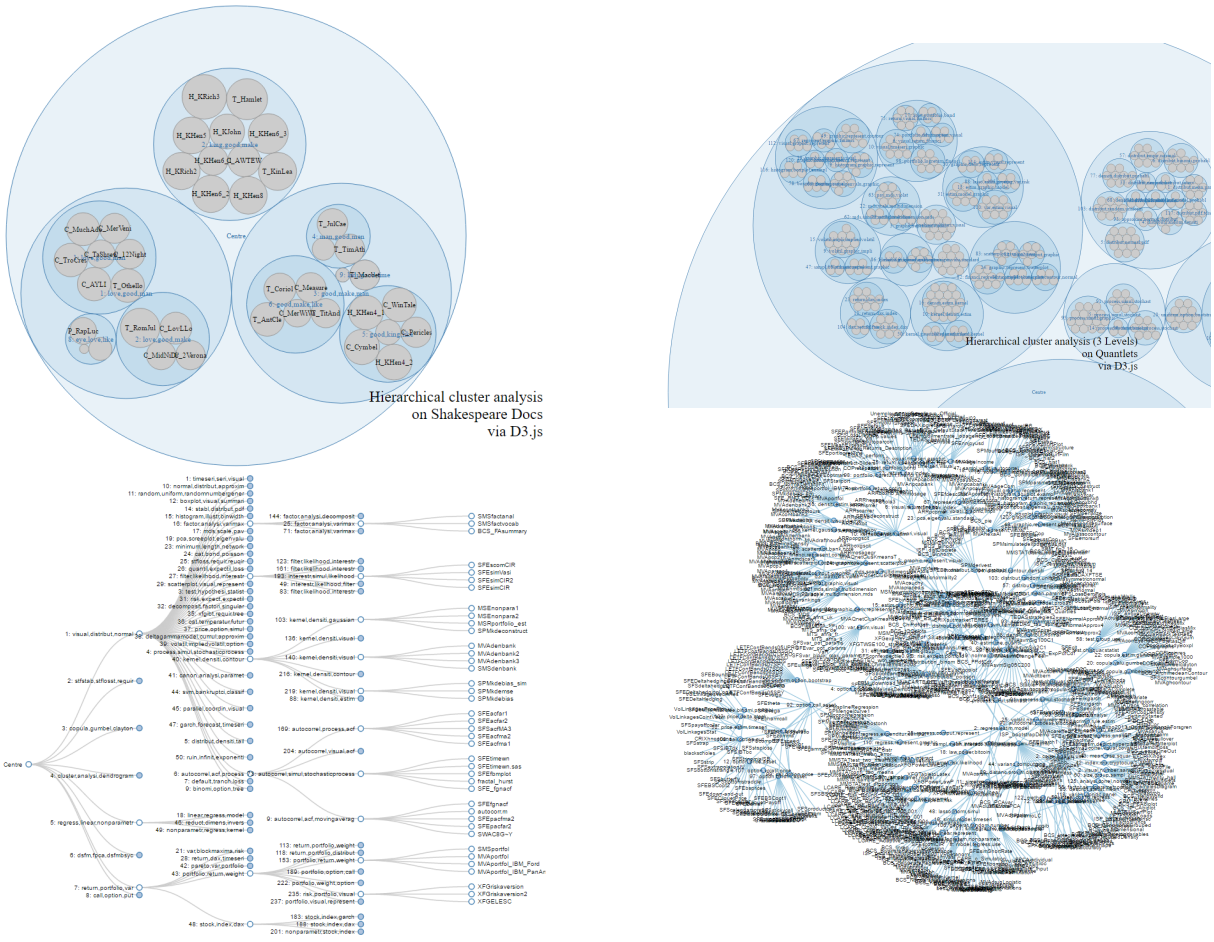


Figure 2.9: 4 Visualization examples from Q3-D3 Genesis Chapters II - VI

Impressed by the performance and universal applicability of the D3 framework, we decided to build the new QuantNet visualization upon the D3 architecture. The first steps are summarized in chapter "I. Genesis (Nov 2013 - Aug 2014)". Basically, all main data objects from QuantNet could be exported to and visualized in the D3 framework templates, amongst them the whole "QuantNet universe" and "galaxies" representing individual subsets like books and projects. Further, co-occurrence information about authors and keywords as well as further details like creation times etc. could be exploited. Not only all source code files from Q3-D3 Genesis are available for free use and reproducibility but also live examples on GitHub pages: `https://github.com/Quantlet/D3Genesis`.

QuantNet contains also all Quantlets (which serve as supplementary examples and exercises) from the following books: MVA (Härdle and Simar, 2015), SFE (Franke et al., 2015), SFS (Borak et al., 2013), XFG (Härdle et al., 2008). These book abbreviations are used in some figures in this section and in Section 2.5.

One of the most popular D3 layouts is the "Force-Directed Graph"[9], which was extensively

[8] `http://vallandingham.me/vis/`
[9] `https://bl.ocks.org/mbostock/4062045`

deployed in the "Genesis" chapter and which is a fundamental part of the final QuantNet visualization. The layout is based on special graph-drawing methods called force-directed techniques. These techniques represent a graph as a system of physical objects with diverse physical forces (e.g. electric) pulling and pushing the objects apart. The optimal visualization layout implies that all these forces are in equilibrium, see for more details Michailidis (2008).

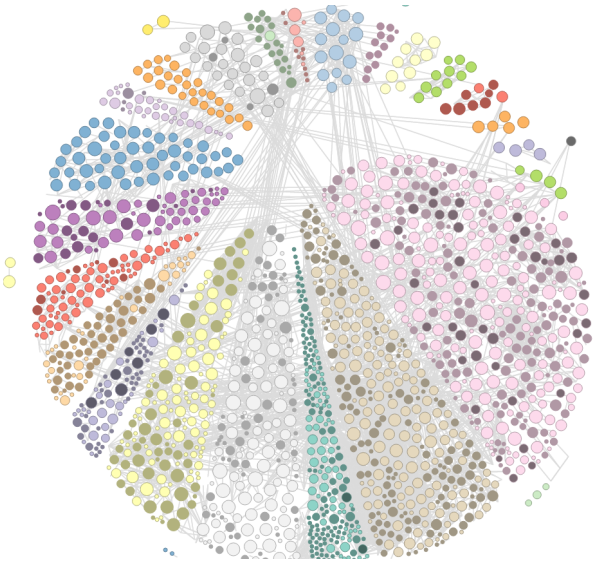


Figure 2.10: Orbit clustering of QuantNet, grouped by books and projects

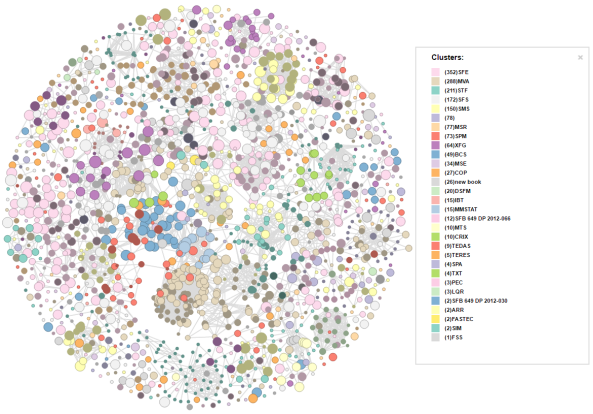


Figure 2.11: Force-Directed Graph of QuantNet, linked by "see also" connections

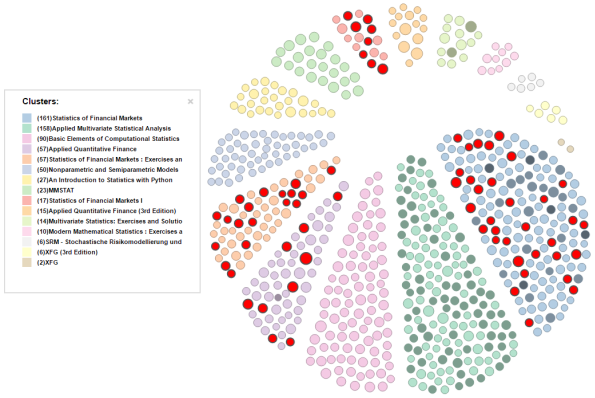


Figure 2.12: Orbit clustering of QuantNet, subset grouped by Springer books. Quantlets containing the search query "black scholes" are highlighted in red

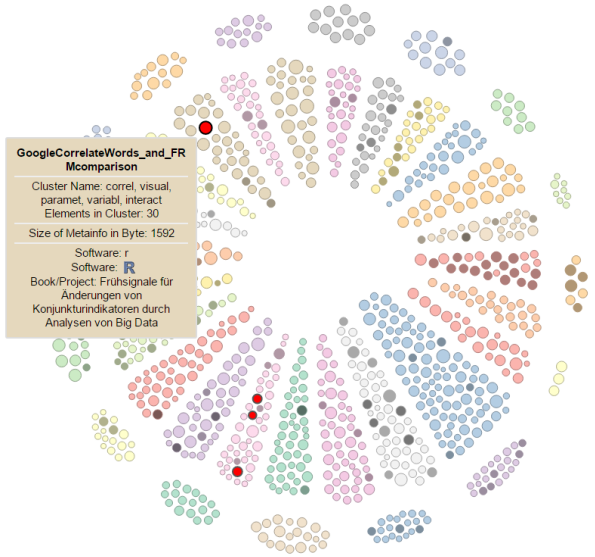


Figure 2.13: Orbit clustering of QuantNet, LSA model, k-means, 40 clusters. Quantlets containing the search query "big data" are highlighted in red

Subsequently, other D3 layouts were examined, which is documented in the chapters from "II. Shakespeare works" to "VI. QuantNet 2.0 @ GitHub". Figure 2.9 shows four visualization examples based on three different D3 layouts: Circle Packing, Force Collapsible with names and Expandable Tree. They are realized via the following D3 classes: `d3.layout.pack`, `d3.layout.force` and `d3.layout.tree`.

Chapter "II. Shakespeare works" served as a simple and impressive example. Further, diverse subsets of QuantNet documents and code files in different stages of development were visualized in five D3 layouts, which are mainly designed for the graphical representation of hierarchically structured data. Specially for this purpose, a dendrogram parser was constructed. Starting with the "document term matrix" of the Quantlets, the R code generated the tree structure and

cluster labels based on the dendrogram which was created by the R function `hclust`. Finally, the recursively structured tree list within R was transformed to a JSON (`http://json.org`) file, which is subsequently required by the D3 designs.

Finally, we see four different examples of the QuantNet Visu from quantlet.de, see Figures 2.10, 2.11, 2.12, 2.13. The TM pipeline retrieves the meta information of Quantlets via the GitHub-R-API, then the LSA model is applied, clusters and labels are generated, and the processed data is transferred via JSON into the D3 Visu application. In the following section the vector space representations (with LSA as a special case of them) will be described in more detail.

2.4 Vector space representations

2.4.1 Text to Vector

The vector space model (VSM) representation for a document d has been introduced by Salton et al. (1975). Given a document, it is possible to associate with it a bag of terms (or bag of words) by simply considering the number of occurrences of all terms contained. Typically words are "stemmed", meaning that the inflection information contained in the last few letters is removed.

A bag of words has its natural representation as a vector in the following way. The number of dimensions is the same as the number of different terms in the corpus, each entry of the vector is indexed by a specific term, and the components of the vector are formed by integer numbers representing the frequency of the term in the given document. Typically such a vector is then mapped/transformed into some other space, where the word frequency information is merged/rescaled considering other information like word importance, relevance and semantic, assigning to uninformative words lower or no weight.

Suppose we have a set of documents Q and a set of terms T. Define $tf(d,t)$ as the absolute frequency of term $t \in T$ in $d \in Q$ and $idf(t) = \log(|Q|/n_t)$ as the inverse document frequency, with $n_t = |\{d \in Q | t \in d\}|$. Let $w(d) = \{w(d,t_1), \ldots, w(d,t_m)\}^\top$, $d \in Q$, be the weighting vector of the given document. Each $w(d,t_i)$ is calculated by a weighting scheme, see next Section 2.4.2. Then $D = [w(d_1), \ldots, w(d_n)]$ is the "term by document" matrix, or in abbreviated form TDM.

In this way a document is represented by a (column) vector $w(d)$, in which each entry reflects the relevance/importance of a particular word stem used in the document. Typically d can have tens of thousands of entries, often more than the number of documents. Furthermore, for a particular document the representation is typically extremely sparse, having only relatively few non-zero entries, more details in Section 2.6.2.

2.4.2 Weighting scheme, Similarity, Distance

A widely used weighting scheme in IR and TM is the tf-idf, short for *term frequency - inverse document frequency*. The concept of *idf* was introduced as "term specificity" by Jones (1972). Although it has worked well as a heuristic, its theoretical foundations have been troublesome for at least three decades afterward, with many researchers trying to find information theoretic justifications for it. Robertson (2004) (who worked from 1998 to 2013 in the Cambridge laboratory of Microsoft Research and contributed to the Microsoft search engine Bing) concludes 32 years later in the same journal "Journal of Documentation":

> However, there is a relatively simple explanation and justification of IDF in the relevance weighting theory of 1976. This extends to a justification of TF*IDF in the Okapi BM25 model of 1994. IDF is simply neither a pure heuristic, nor the theoretical mystery many have made it out to be. We have a pretty good idea why it works as well as it does.

The (normalized) *tf-idf* weighting scheme is defined as

$$w(d,t) = \frac{tf(d,t)idf(t)}{\sqrt{\sum_{j=1}^{m} tf(d,t_j)^2 idf(t_j)^2}}, m = |T|. \tag{2.1}$$

Hence, the similarity of two documents d_1 and d_2 (or the similarity of a document and a query vector q) can be computed based on the inner product of the vectors. The (normalized *tf-idf*) similarity S of two documents d_1 and d_2 is given by

$$S(d_1,d_2) = \sum_{k=1}^{m} w(d_1,t_k) \cdot w(d_2,t_k) = w(d_1)^\top w(d_2). \tag{2.2}$$

A frequently used distance measure is the Euclidean distance:

$$dist_2(d_1,d_2) = \sqrt{\sum_{k=1}^{m} \{w(d_1,t_k) - w(d_2,t_k)\}^2}. \tag{2.3}$$

It holds the general relationship:

$$\cos\phi = \frac{x^\top y}{|x| \cdot |y|} = 1 - \frac{1}{2} dist^2 \left(\frac{x}{|x|}, \frac{y}{|y|} \right), \tag{2.4}$$

with ϕ as the angle between x and y. Substituting $\frac{x}{|x|}$ by $w(d_1)$ and $\frac{y}{|y|}$ by $w(d_2)$, we have an easily computable transformation between the *tf-idf* similarity and the Euclidean distance. In particular when dealing with big data this fact can be exploited, since many standard clustering methods expect a distance matrix in advance. Usually, it is more efficient to first calculate the similarity matrix, exploiting the strong sparsity in text documents, and then apply the transformation in Formula 2.4 to obtain the distance matrix.

2.4.3 Shakespeare's tragedies

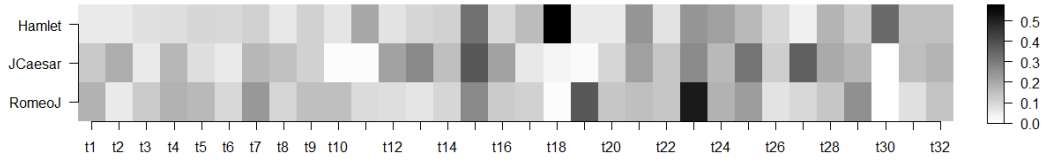


Figure 2.14: Heatmap of T_s in 3 Shakespeare's tragedies

The basic concepts of the introduced vector space representations will be illustrated by the example of Shakespeare's works, available under `http://shakespeare.mit.edu`. Let $Q = \{d_1, d_2, d_3\}$ be the document corpus containing the following Shakespeare's tragedies: d_1 = "Hamlet" (total word number: 16769); d_2 = "Julius Caesar" (total word number: 11003); d_3 = "Romeo and Juliet" (total word number: 14237). After some text preprocessing as in Section 2.6.1, the TDM is a 5521×3 matrix. Consider the special vocabulary T_s, selected amongst 100

most frequent words:

$$\begin{aligned} T_s &= \{art, bear, call, day, dead, dear, death, die, eye, fair, father, fear, \\ &\quad friend, god, good, heart, heaven, king, ladi, lie, like, live, love, \\ &\quad make, man, mean, men, must, night, queen, think, time\} \\ &= \{t_1, \ldots, t_{32}\} \end{aligned}$$

Figure 2.16 shows the weighting vectors $w(d)$ of the tragedies in Q (Hamlet, Julius Caesar, Romeo and Juliet) wrt. to the special vocabulary T_s in a radar chart. The highest term weightings $w(d,t)$ are distributed as follows: $w(d_1, t_{18})$, $t_{18} \hat{=}$ "king"; $w(d_1, t_{30})$, $t_{30} \hat{=}$ "queen"; $w(d_2, t_{15})$, $t_{15} \hat{=}$ "good"; $w(d_2, t_{27})$, $t_{27} \hat{=}$ "men"; $w(d_3, t_{19})$, $t_{19} \hat{=}$ "ladi"; $w(d_3, t_{23})$, $t_{23} \hat{=}$ "love". The heatmap in Figure 2.14 displays the same information in another representation.

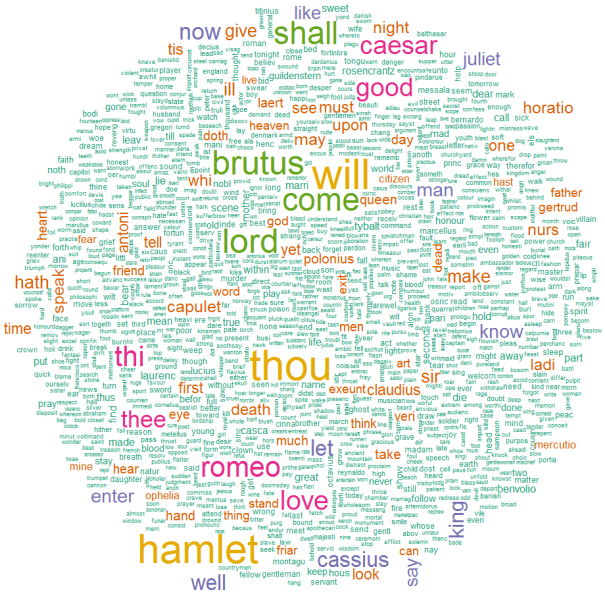


Figure 2.15: Wordcloud of all words (tf $\geq$ 5) in 3 Shakespeare's tragedies in corpus Q

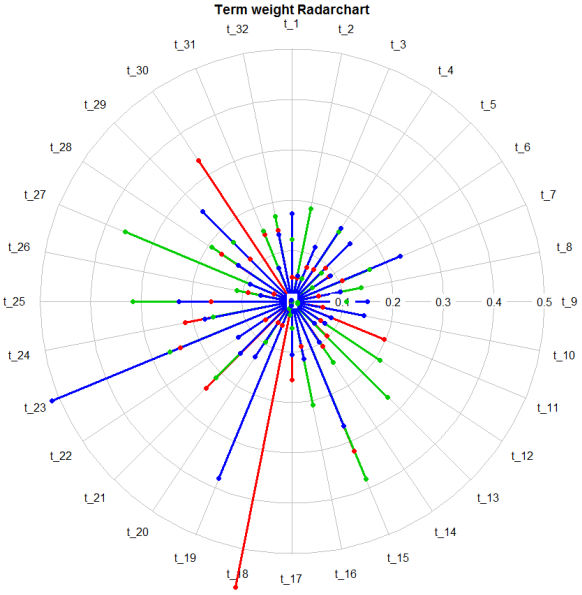


Figure 2.16: Radar chart: weightings of terms in T_s of tragedies in corpus Q.

M_S and M_D for 32 special terms in T_s:

$$M_S = \begin{pmatrix} 1 & 0.64 & 0.63 \\ 0.64 & 1 & 0.77 \\ 0.63 & 0.77 & 1 \end{pmatrix} \quad M_D = \begin{pmatrix} 0 & 0.85 & 0.87 \\ 0.85 & 0 & 0.68 \\ 0.87 & 0.68 & 0 \end{pmatrix}$$

M_S and M_D for all 5521 terms:

$$M_S = \begin{pmatrix} 1 & 0.39 & 0.46 \\ 0.39 & 1 & 0.42 \\ 0.46 & 0.42 & 1 \end{pmatrix} \quad M_D = \begin{pmatrix} 0 & 1.10 & 1.04 \\ 1.10 & 0 & 1.07 \\ 1.04 & 1.07 & 0 \end{pmatrix}$$

Finally, we present the similarity matrices M_S and distance matrices M_D for the selected tragedies in Q. On the one hand, wrt. to the special vocabulary T_s, on the other hand, wrt. to the full vocabulary containing 5521 terms. Every entry in M_S and M_D corresponds to the value calculated by Formula 2.2 and 2.3, respectively, for any given document pair $d_i, d_j \in Q$. The

weighting scheme was calculated via the normalized tf weight. In the case of a few documents in the corpus the document frequency *idf* is inappropriate as many frequent terms have a high probability to be present in all documents, in this case only three. Therefore, the *idf* weighting share would make many terms vanish, which would considerably decrease the overall similarity between two documents which is calculated by the scalar product of their term weights.

2.4.4 Generalized VSM (GVSM)

One of the problems with basic VSM representations as presented in Section 2.4.1 is that they treat terms as uncorrelated, assigning them into orthogonal directions in the feature space. A classical example is synonymous words which contain the same information, but are assigned distinct components (Srivastava and Sahami, 2009). As a consequence, only documents that share many terms (which serve as vector components) can be clustered into common topics and clusters. But in reality words are correlated, and sometimes even synonymous, so that documents with very few common terms can potentially be on closely related topics. Such similarities cannot be detected by the basic vector space model (BVSM) (Salton et al., 1975). This raises the question of how to incorporate information about semantics into the feature map, so as to link documents that share "related" terms?

So far, we have identified the following drawbacks of the classical tf-*idf* approach and of the BVSM in general: 1) uncorrelated/orthogonal terms in the feature space, 2) documents must have common terms to be similar, 3) sparsity of document vectors and similarity matrices.

Over the time many solutions were proposed by various researchers, first of them Wong et al. (1985) and Deerwester et al. (1990). We will treat them later in this section. Other noteworthy books giving a general survey of the big topic "Text mining and different models" are (Berry, 2003) and (Srivastava and Sahami, 2009). The most popular solutions are: I) using statistical information about term-term correlations (GVSM in Section 2.4.4); II) incorporating information about semantics (semantic smoothing, LSA in Section 2.4.4).

More generally, we can consider transformations of the document vectors by some mapping P. The simplest case involves linear transformations, where P is any appropriately shaped matrix. In this case the generalized similarity S has the form:

$$S_P(d_1, d_2) = (Pd_1)^\top (Pd_2) = d_1^\top P^\top P d_2, \quad d_1, d_2 \in Q. \tag{2.5}$$

Every P defines another generalized vector space model (GVSM), resulting in the similarity matrix:

$$M_S^{(P)} = D^\top (P^\top P) D,$$

with D being the "term by document" matrix as defined in Section 2.4.1.

Basic VSM (BVSM)

The BVSM was introduced by Salton et al. (1975) and uses the vector representation with no further mapping, the VSM shows $P = I$ in this case. Even in this simple case the "matrix nature" of VSM allows different embeddings of tf-*idf* weightings into the matrix representations.

- $P = I_m$ and $w(d) = \{tf(d, t_1), \ldots, tf(d, t_m)\}^\top$ lead to the classical tf-similarity $M_S^{tf} = D^\top D$
- diagonal $P(i, i)^{idf} = idf(t_i)$ and $w(d) = \{tf(d, t_1), \ldots, tf(d, t_m)\}^\top$ lead to the classical tf-*idf*-similarity $M_S^{tfidf} = D^\top (P^{idf})^\top P^{idf} D$
- starting with $w(d) = \{tf(d, t_1)idf(t_1), \ldots, tf(d, t_m)idf(t_m)\}^\top$ and $P = I_m$ results in the classical tf-*idf*-similarity $M_S^{tfidf} = D^\top I_m D = D^\top D$ as well

GVSM – term-term correlations

An early attempt to overcome the limitations of the BVSM was proposed by Wong et al. (1985) under the name of generalized VSM, or GVSM. A document is characterized by its relation to other documents in the corpus as measured by the BVSM. The mapping P and the resulting model specifications are as follows:

- $P = D^\top$ is the linear mapping
- $S(d_1, d_2) = (D^\top d_1)^\top (D^\top d_2) = d_1^\top DD^\top d_2$ is the document similarity
- $M_S^{TT} = D^\top (DD^\top) D$ is the similarity matrix

$DD^\top$ is called a "term by term" matrix, having a nonzero ij entry if and only if there is a document containing both the i-th and the j-th term. Thus, terms become semantically related if they co-occur often in the same documents. The documents are mapped into a feature space indexed by the documents in the corpus, as each document is represented by its relation to the other documents in the corpus. If the BVSM represents a document as bag of words, the GSVM represents a document as a vector of its similarities relative to the different documents in the corpus. If there are less documents than terms, then we additionally achieve a dimensionality reduction effect. In order to avoid misleading we will refer to this model as the GVSM(TT) for the rest of our article, hence distinguishing it from other possible GVSM representations which are induced by another mapping P.

GVSM – Latent Semantic Analysis (LSA)

Latent semantic analysis (LSA) is a technique to incorporate semantic information in the measure of similarity between two documents (Deerwester et al., 1990). LSA measures semantic information through co-occurrence analysis in the corpus. The document feature vectors are projected into the subspace spanned by the first k singular vectors of the feature space. The projection is performed by computing the singular value decomposition (SVD) of the matrix $D = U\Sigma V^\top$. Hence, the dimension of the feature space is reduced to k and we can control this dimension by varying k. This is achieved by constructing a modified (or truncated) matrix D_k from the k-largest singular values σ_i, $i = 1, 2, 3, ..., k$, and their corresponding vectors: $D_k = U_k \Sigma_k V_k^\top$. Based on the SVD factors, the resulting model specifications are as follows:

- $P = U_k^\top := I_k U^\top$ is the projection operator onto the first k dimensions, I_k is a $m \times m$ identity matrix having ones only in the first k diagonal entries, $k < m$
- $M_S^{LSA} = D^\top (U I_k U^\top) D$ is the similarity matrix
- $D_k = UPD = U_k \Sigma_k V_k^\top = U \Sigma_k V^\top$ is the truncated TDM which is re-embedded into the original feature space, $PD = \Sigma_k V^\top$ is the corresponding counterpart in the semantic space
- $D_{err} = D - D_k = U(\Sigma - \Sigma_k) V^\top$ is the approximation error of the SVD truncation

The k dimensions can be interpreted as the main semantic components/concepts and $U_k U_k^\top = U I_k U^\top$ as their correlation. Some authors refer to $U I_k U^\top$ as a "semantic kernel" or "latent semantic kernel". It can be shown that $M_S^{LSA} = V \Lambda_k V^\top$. Starting with $V \Lambda V^\top = V \Sigma^\top \Sigma V^\top = V \Sigma^\top U^\top U \Sigma V^\top = D^\top D$ and diagonal $\Lambda_{ii} = \lambda_i = \sigma_i^2$ with eigenvalues of $D^\top D$, the truncated diagonal Λ_k consists of the first k eigenvalues and zero-values else. It should be noted that $D^\top D$ is the BVSM similarity matrix. For more technical and scientific proofs and interpretations of this paragraph we recommend the following publications: Cristianini et al. (2002), Berry (2003) and Srivastava and Sahami (2009). The visualization of the "LSA anatomy" in Section 2.6.4 may also be helpful.

Closer look at the LSA implementation

Several classes of adjustment parameters can be functionally differentiated in the LSA process. Every class introduces new parameter settings that drive the effectiveness of the algorithm. The following classes have been identified so far by Wild and Stahl (2007):

1. Textbase compilation and selection
2. Preprocessing: stemming, stopword filtering, special vocabulary etc.
3. Weighting schemes: local weights (none (i.e. tf), binary tf, log tf etc.); global weights (normalisation, idf, entropy etc.)
4. Dimensionality: singular values k (coverage of total weight = 0.3, 0.4, 0.5 etc.)
5. Similarity measurement: cosine, best hit, mean of best, pearson, spearman etc.

The latent semantic space can be either created directly by using the documents, in our case Quantlets, letting the matrix D be the weighting vectors of the Quantlets. Or it can be first trained by domain-specific and generic background documents. Generic texts add thereby a reasonably heterogeneous amount of general vocabulary, whereas the domain-specific texts provide the professional vocabulary. The Quantlets would be then folded in into the semantic space which was created in the previous SVD process. By doing so, one gains in general a higher retrieval performance as the vocabulary set is bigger and more semantic structure is embedded.

Bradford (2009) presented an overview of 30 sets of studies in which the LSA performance in text processing tasks could be compared directly to human performance on the same tasks. In half of the studies, performance of LSA was equal to or better than that of humans.

Miller et al. (2009) proposed a family of LSA-based search algorithms which is designed to take advantage of the semantic properties of well-styled hyperlinked texts such as wikis. Performance was measured by having human judges rating the relevance of the top four search results returned by the system. When given singleterm queries, the highest-performing search algorithm performed as well as the proprietary PageRank-based Google search engine. The comparison with respect to Google is especially promising, given that the presented system operated on less than 1% of the original corpus text, whereas Google uses not only the entire corpus text but also meta data internal and external to the corpus.

Fernández-Luna et al. (2011) proposed a recommender agent based on LSA formalism to assist the users that search alone to find and join to groups with similar information needs. With this mechanism, a user can easily change her solo search intent to explicit collaborative search.

A comparison of three WordNet related methods for taxonomic-based sentence semantic relatedness was examined in Mohamed and Oussalah (2014). Using a human annotated benchmark data set, all three approaches achieved a high positive correlation, reaching up to $r = 0.88$ with comparison to human ratings. In parallel, two other baseline methods (LSA as part of it) evaluated on the same benchmark data set. LSA showed comparable correlation as the more sophisticated WordNet (`https://wordnet.princeton.edu`) based methods.

GVSM Applicability for Big Data

Having n documents with a vocabulary of m terms and LSA truncation to k dimensions, there are the following memory space requirements for the TDM representations: $m \times n$ matrix cells in BVSM ($\mathcal{O}(mn)$); n^2 matrix cells in GVSM(TT) ($\mathcal{O}(n^2)$); $k \times (k + m + n)$ matrix cells in LSA(k) ($\mathcal{O}(kn)$). In the context of big data the n will usually dominate the other quantities m and k, furthermore k is fixed, see for comparison Table 2.1. Clearly, the TDM D in the BVSM is the first step for all three models. Hence, the basic calculation and storage demand is dictated by D. Concerning the memory demands, the GVSM(TT)-TDM would be maximal. For a fixed k the memory demand for a TDM in LSA would be less than in BVSM: $\mathcal{O}(kn)$ versus $\mathcal{O}(mn)$. The calculation of the GVSM(TT)-TDM would involve a matrix multiplication $D^\top D$, implying

$n^2 \times m$ multiplications. Concerning the LSA, which is performed by SVD, the situation is more complex.

There are numerous theoretical approaches and software implementations with respect to the SVD topic. Several state-of-the-art algorithms including the Lanczos-based truncated SVD and the corresponding implementations are outlined in Korobeynikov (2010) and Golyandina and Korobeynikov (2014). The R package **svd** (Korobeynikov et al., 2016) provides "Interfaces to Various State-of-Art SVD and Eigensolvers" (`https://github.com/asl/svd`). This package is basically an R interface to the software package PROPACK[10] containing a set of functions for computing the singular value decomposition of large and sparse or structured matrices, which are written in Fortran and C. Although the R package **lsa** (Wild, 2015), which performs a full SVD, is sufficient for the QuantNet data, some benchmarks were run applying the function `propack.svd` from the R package **svd** to examine its performance. The main advantages are the time saving partial SVD calculation (depending on k) and the fast C optimized implementation. For this purpose I have extracted several data sets from GitHub by means of the "GitHub Mining Infrastructure in R" (see Section 2.8.1). The collected data are meta informations describing samples of GitHub organizations.

time in seconds for	BVSM	LSA(k)	BVSM + LSA(k)	size of TDM (BVSM)
10.570 Org's	39	149	188	14238×10570
16.803 Org's	51	264	315	16029×16803
30.437 Org's	69	637	706	18501×30437
45.669 Org's	93	990	1083	20368×45669
97.444 Org's	159	2673	2832	23667×97444

Table 2.1: Benchmark for TDM matrix creation in BVSM (package **tm**) and LSA(k) (propack.svd from package **svd**), $k = 100$, elapsed time in seconds

As can be inferred from Table 2.1, the time complexity both for BVSM and LSA TDM matrix creation is feasible. 10570 data sets from GitHub organizations require less than 1 minute for BVSM and two and a half minutes for LSA. Increasing the number of data up to roughly 100000 samples leads to less than 3 minutes calculation time for BVSM and 45 minutes for LSA. In simpler terms, one can create a TDM for 100.000 documents both in BVSM and LSA in less than one hour on a single CPU core without any parallelization expense. A smaller data set like 10.000 documents can be handled on an usual PC with 8 GByte RAM. For larger data sets a Linux server (Research Data Center, `https://rdc.hu-berlin.de`) with an available memory of 256 GiB was used. Since this benchmark was focused on the time complexity, no deeper analysis was undertaken concerning the memory demand. At any time point of the benchmark process the available RAM of 256 GiB was far away from being exhausted.

Concluding we can say that a Linux server with 256 GiB RAM has sufficient performance reserves for BVSM and LSA processing of big data, having 100.000 documents and an hour processing time as a "lower boundary". As software one needs only an R installation and some freely available R packages (**tm**, **svd** as the most crucial ones). All tests were conducted on a single core, hence there is additional potential to speed up the calculation time. In Theußl et al. (2012) a **tm** plug-in called **tm.plugin.dc** is presented implementing a distributed corpus class which can take advantage of the Hadoop MapReduce library for large scale text mining tasks. With a quadratic space complexity (memory demand) of $\mathcal{O}(n^2)$ and a cubic time complexity of $n^2 \times m$ multiplications, the GVSM(TT) model is the worst choice among the considered TM models, unless some optimization (like parallelization, exploiting theoretical properties like sparsity etc.) is done.

[10] `http://sun.stanford.edu/~rmunk/PROPACK/`

2.5 Methods

2.5.1 Cluster Analysis

If the data can validly be summarized by a small number of groups of objects, then the group labels may provide a very concise description of patterns of similarities and differences in the data. The need to summarize data sets in this way is increasingly important because of the growing volumes of data now available in many areas of science, and the exploration process of such data sets using cluster analysis and other multivariate analysis techniques is now often called data mining. In the 21st century, data mining has become of particular interest for investigating material on the World Wide Web, where the aim is to gather and analyze useful information or knowledge from web page contents (Everitt et al., 2011).

Our objectives are to determine topic labels and assign them to (text) documents. A confident and reliable automatic process would completely bypass the expense of having humans, whose task is to provide labels. But the process known as document clustering is less than perfect. The labels and their assignment may vary depending on humans or different objective processes that incorporate external information such as stock price change. Document clustering assigns each of the documents in a collection to one or more smaller groups called clusters (Weiss et al., 2010).

The result of clustering is typically a partition (also called clustering) $\mathcal{C}$, a set of clusters C. Each cluster/group consists of a number of documents d. Objects - in our case documents - of a cluster should be similar within the same group and dissimilar to documents of other groups (Hastie et al., 2009). The code for the reproducibility of the clustering in Figure 2.18 is available as interactive MVAQnetClusKmeans_plotly.

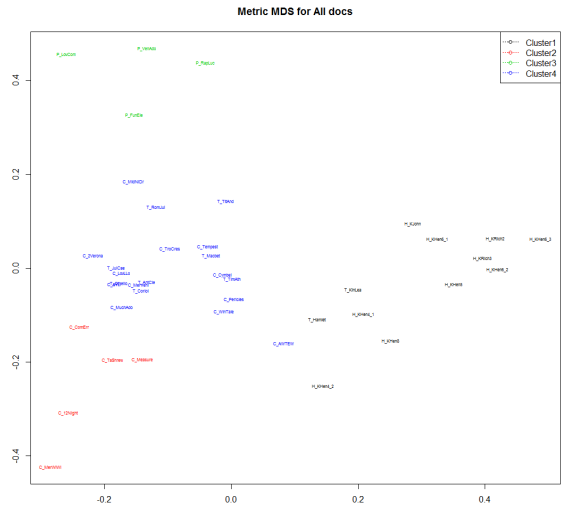


Figure 2.17: k-means clustering of Shakespeare's works

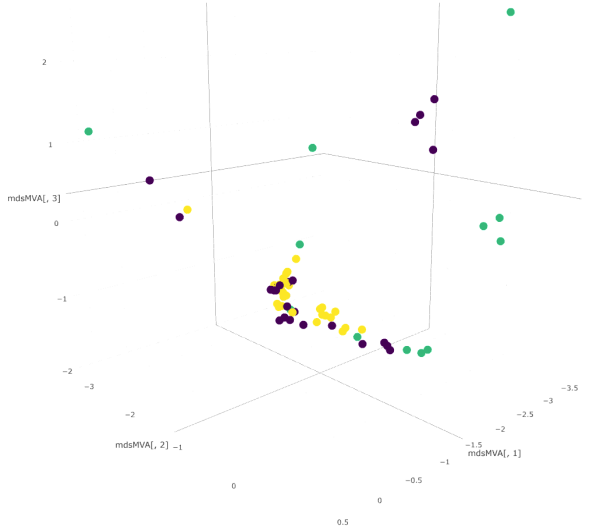


Figure 2.18: k-means clustering and metric MDS for MVA quantlets via Plotly

Partitional Clustering

k-means is a classical clustering method that has been adapted to documents. It is very widely used for document clustering and is relatively efficient. The k-means algorithm aims to partition n observations/objects into k clusters in which each observation is assigned to the cluster with the nearest mean, serving as a prototype of the cluster. k-means typically converges to its

minimum after relatively few iterations.

k-medoids clustering is related to the k-means. It is also referred to as partitioning around medoids or PAM. Both variants attempt to minimize the distance between points labeled to be in a cluster and a point designated as the center/medoid of that cluster. In contrast to the k-means, k-medoids chooses datapoints as centers and works with an arbitrary matrix of distances. Concerning their R implementations `kmeans` and `pam`, the function `pam` is more robust because it minimizes a sum of unsquared dissimilarities. Moreover, `pam` does not need initial guesses for the cluster centers, contrary to `kmeans` (Kaufman and Rousseeuw, 2008).

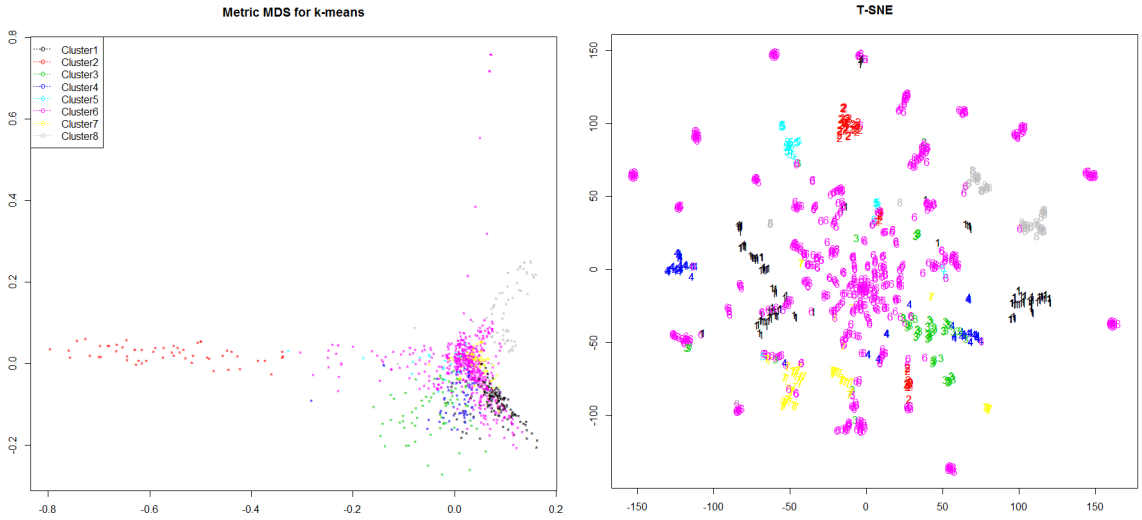


Figure 2.19: LSA:50 geometry of Quantlets via MDS (left) and t-SNE (right), clustered by k-means with generated topics

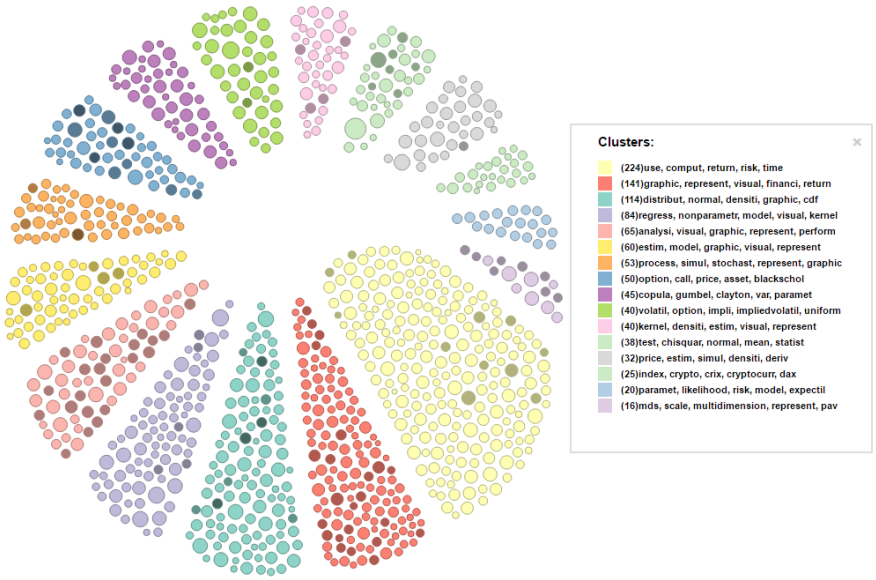


Figure 2.20: Quantlets clustered by k-means into 16 clusters, the tooltip on the right shows their topics

In Figure 2.19 `kmeans` produced 8 clusters with the following topic assignments: 1) "distribut copula normal gumbel pdf"; 2) "call option blackschol put price"; 3) "return timeseri dax stock

financi"; 4) "portfolio var pareto return risk"; 5) "interestr filter likelihood cir term"; 6) "visual dsfm requir kernel test"; 7) "regress nonparametr linear logit lasso"; 8) "cluster analysi pca principalcompon dendrogram". The cluster topics were created based on the most frequent terms of cluster centroids. A multidimensional scaling (MDS) output of the `pam` function with cluster labeling can be reproduced by YAMLcentroids.

Hierarchical Clustering

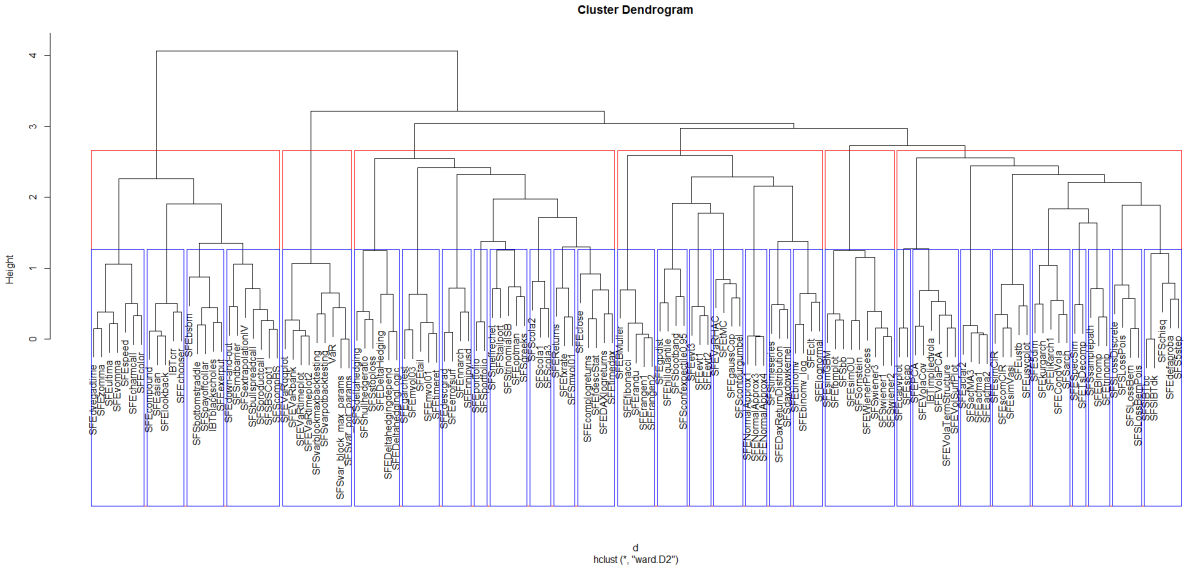


Figure 2.21: Dendrogram created by HC (ward-method) in LSA model, cut in 6 clusters and 30 subclusters, 137 Gestalten, subset from the books SFE, SFS and the project IBT

Hierarchical clustering algorithms got their name since they form a sequence of groupings or clusters that can be represented in a hierarchy of nested clusters (Steinbach et al., 2000). This hierarchy can be obtained either in a top-down or bottom-up fashion. Top-down means that we start with one cluster that contains all documents. This cluster is stepwise refined by splitting it iteratively into sub-clusters. One speaks in this case also of the so called "divisive" algorithm. The bottom-up or "agglomerative" procedures start by considering every document as an individual cluster. Then the most similar clusters are iteratively merged, until all documents are contained in one single cluster. In practice the divisive procedure is almost of no importance due to its generally bad results. Therefore, only the agglomerative variants are outlined in the following. Typical agglomeration methods are "ward.D", "ward.D2", "single", "complete" and "average". This family of agglomeration methods will be abbreviated as HC in the following, all of them are available by means of the R function `hclust`.

Hierarchical (agglomerative) clustering is a popular alternative to k-means clustering of documents. As explained above, the method produces clusters, but they are organized in a hierarchy comparable with a table of contents for a book. The binary tree produced by HC is a map of many potential groupings of clusters. One can process this map to get an appropriate number of clusters. That is more difficult with k-means, where the procedure usually must be restarted when we specify a new value of k.

Hierarchical classifications produced by either the agglomerative or divisive route may be represented by a two-dimensional diagram known as a *dendrogram*, which illustrates the fusions

or divisions made at each stage of the analysis. Two examples of such a dendrogram are given in Figures 2.21 and 2.22.

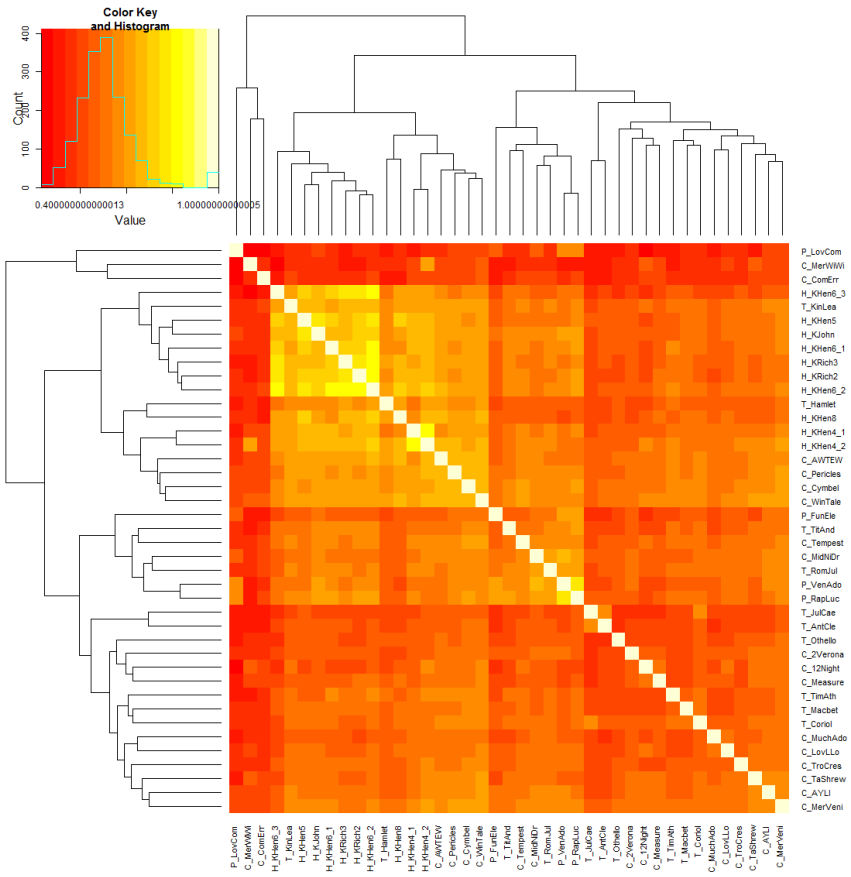


Figure 2.22: Combined representation of Shakespeare's works: their similarity matrix via heat map, histogram of the matrix values and dendrograms of the row and column values (created via heatmap.2 function from the R package **gplots**)

2.5.2 Cluster validation measures

Internal validation measures take only the data set and the clustering partition as input and use intrinsic information in the data to assess the quality of the clustering. For internal validation, we decided for measures that reflect the *compactness*, *connectedness* and *separation* of the cluster partitions. Connectedness relates to what extent observations are placed in the same cluster as their nearest neighbours in the data space, and is measured by the *connectivity* method as suggested by Handl et al. (2005). Compactness assesses cluster homogeneity, usually by looking at the intra-cluster variance, while separation quantifies the degree of separation between clusters, usually by measuring the distance between cluster centroids. Since compactness and separation demonstrate opposing trends (compactness increases with the number of clusters but separation decreases), popular methods combine the two measures into a single score. The *Dunn Index* (Dunn, 1974) and *Silhouette Width* (Rousseeuw, 1987) are both examples of non-linear combinations of the compactness and separation. Together with the connectivity method they constitute the three internal measures available in the R package **clValid** (Brock et al., 2008). The details of each measure are given below, and for a good overview of internal measures in general see Handl et al. (2005).

Connectivity

The *connectivity* indicates the degree of connectedness of the clusters, as determined by the k-nearest neighbours. Let N denote the total number of observations (documents) in a data set. Define $nn_{i(j)}$ as the jth nearest neighbour of observation i, and let $x_{i,nn_{i(j)}}$ be zero if i and j are in the same cluster and $1/j$ otherwise. Then, for a particular clustering partition $\mathcal{C} = \{C_1, \ldots, C_K\}$ of the N observations into K disjoint clusters, the connectivity is defined as

$$Conn(\mathcal{C}) = \sum_{i=1}^{N} \sum_{j=1}^{L} x_{i,nn_{i(j)}} \ , \tag{2.6}$$

where L is a parameter giving the number of nearest neighbours to use. The connectivity has a value between zero and ∞ and should be minimized.

Silhouette

The *Silhouette* of a datum is a measure of how closely it is matched to data within its cluster and how loosely it is matched to data of the neighbouring cluster, i.e. the cluster whose average distance from the datum is lowest. A Silhouette close to 1 implies the datum is in an appropriate cluster, while a Silhouette close to -1 implies the datum is in the wrong cluster. For observation i, it is defined as

$$S(i) = \frac{b_i - a_i}{max(b_i, a_i)} \, , \tag{2.7}$$

where a_i is the average distance between i and all other observations in the same cluster, and b_i is the average distance between i and the observations in the "nearest neighbouring cluster", i.e.

$$b_i = \min_{C_k \in \mathcal{C} \setminus C(i)} \sum_{j \in C_k} \frac{dist(i,j)}{n(C_k)} \, , \tag{2.8}$$

where $C(i)$ is the cluster containing observation i, $dist(i,j)$ is the distance (e.g. Euclidean, Manhattan) between observations i and j, and $n(C)$ is the cardinality of cluster C. The Silhouette Width is the average of each observation's Silhouette value:

$$Silh(\mathcal{C}) = \frac{1}{N} \sum_{i=1}^{N} \ S(i) \ . \tag{2.9}$$

The Silhouette Width thus lies in the interval $[-1, 1]$, and should be maximized. For more information, see the help page for the `silhouette` function in the package **cluster** (Maechler et al., 2016).

Dunn Index

The *Dunn Index* is the ratio of the smallest distance between observations not in the same cluster to the largest intra-cluster distance. It is computed as

$$Dunn(\mathcal{C}) = \frac{\min\limits_{C_k, C_l \in \mathcal{C}, C_k \neq C_l} \left(\min\limits_{i \in C_k, j \in C_l} dist(i,j) \right)}{\max\limits_{C_m \in \mathcal{C}} diam(C_m)} \, , \tag{2.10}$$

where $diam(C_m)$ is the maximum distance between observations in cluster C_m. The Dunn Index has a value between zero and ∞, and should be maximized.

2.5.3 Visual cluster validation

As long as the data set remains limited and the topic number is of modest size, cluster validation can be easily conducted using visual inspection of the generated topics and the resulting cluster content, comparing them with prior domain specific knowledge. Figure 2.23 demonstrates that through the example of the Quantlets belonging to the book SFE : "Statistics of Financial Markets" (Franke et al., 2015). Incorporating the domain knowledge of the SFE book, the dominating first 8 clusters/topics (corresponding to 96% of the data set) deal with "stochastic process simulation", "returns", "dax", "financial stocks", "call option prices", "assets", "black scholes", "normal distribution density", "probability", "parameter computation", "simulation", "correlation", "model estimation", "finance", "options", "implied volatility". The cluster topics are displayed in the cluster legend on the right in Figure 2.23. The remaining four topics (corresponding to 4% of the data set) also show good concordance with the appropriate cluster content like "kernel density estimation", "nonparametric regression", "risk" etc. Since the automatically generated topic labels consist of stemmed words, the listed "human readable versions" were syntactically improved for illustration purpose by the authors, see also Section 2.8.2.

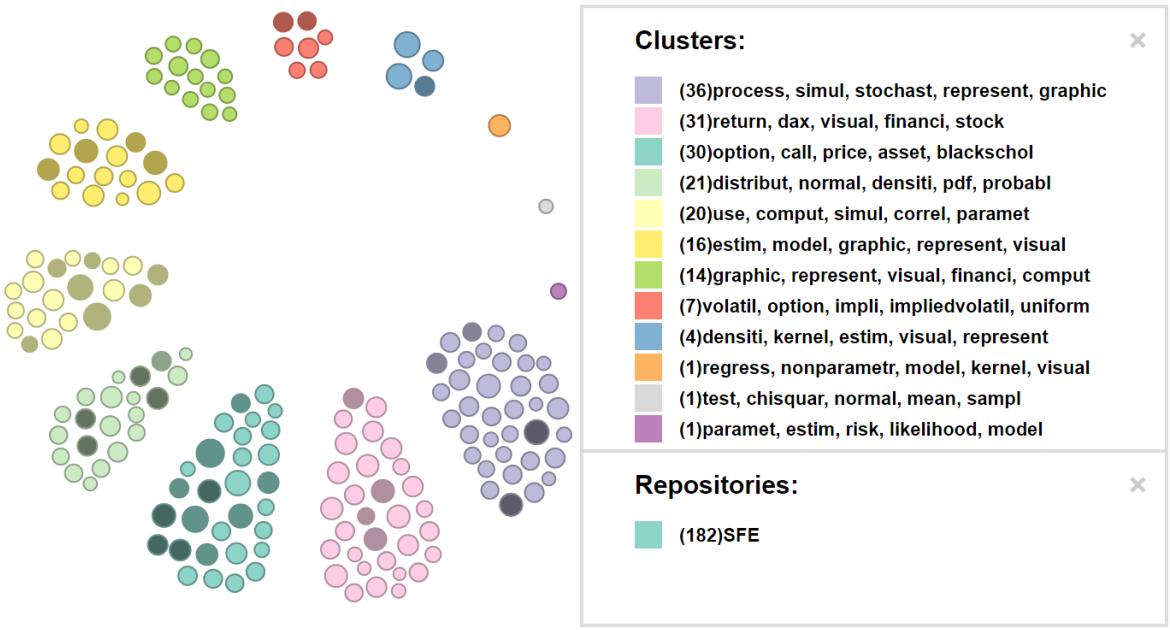
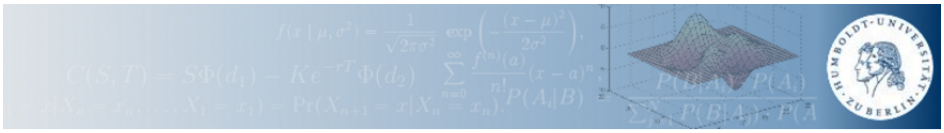


Figure 2.23: SFE Quantlets clustered by k-means into 12 clusters, the tooltip on the right shows their topics

2.6 Results

As data set for the following examination and analysis the whole QuantNet data base was taken. At the time of the big data analysis the documents structure was as follows: 1170 Gestalten (from 1826 individual Quantlets). That means that the meta information was extracted from Quantlets, in the case that several Quantlet versions in different programming languages were available, their meta information was merged to a single and unique representation, called "Gestalt". SFEGBMProcess is such an example, see Figure 2.24.



SFEGBMProcess

```
Name of QuantLet : SFEGBMProcess

Published in : Statistics of Financial Markets

Description : 'Simulation of discrete observations of a Geometric Brownian Motion (GBM) via direct
integration (method=1) or Euler scheme (method=2). The process follows the stochastic differential
equation: dX(t) = mu X(t) dt + sigma X(t) dW(t).'

Keywords : 'Euler, brownian-motion, direct integration, discrete, geometric-brownian-motion,
graphical representation, normal-distribution, plot, process, random-number-generation, simulation,
stochastic, stochastic-process, wiener-process'

See also : SFEsimGBM, SFEbsbm, SFEDeltaHedging, SFEWienerProcess, SFSbb, SFEwienerdens, SFEsimCIR

Author : Alexander Ristig

Author[Matlab] : Ekaterina Ignatieva

Author[SAS] : Daniel T. Pele

Submitted : Tue, June 17 2014 by Franziska Schulz

Input:
- method: 'type of method used: 1 - direct integration, 2 - Euler scheme'

Example : 'A plot of typical path of a geometric brownian motion is provided for the case n=1,
x0=0.084, mu=0.02, sigma=sqrt(0.1) delta=1/1000.'
```

Figure 2.24: Gestalt "SFEGBMProcess" simulating the geometric Brownian motion comprises 3 Quantlets in 3 programming languages: R, Matlab and SAS

Throughout the whole Section 2.6 we will use the definitions and notations from Section 2.4 and 2.5. The first step is to transform the text documents into the quantities listed below. This will be demonstrated in Section 2.6.1.

- $Q = \{d_1, \ldots, d_n\}$: set of documents (Quantlets/Gestalten)
- $T = \{t_1, \ldots, t_m\}$: dictionary (set of all terms)
- $tf(d, t)$: absolute frequency of term $t \in T$ in $d \in Q$
- $D = [w(d_1), \ldots, w(d_n)]$: "term by document" matrix TDM

2.6.1 Text Preprocessing results

For the basic text preprocessing and calculation of the TDM the R package **tm** (Feinerer and Hornik, 2015) was applied, see Listing 2.1. It provides a framework for text mining applications within R (Feinerer et al., 2008). According to Table 2.2 we selected the preprocessing configuration "discarding tf ≤ 2", resulting in a TDM with 1039×1170 entries.

Listing 2.1: Text preprocessing via the **tm** R package

```
# preprocessing text with this function
cleanCorpus = function(corpus) {
        corpus.tmp <- tm_map(corpus, removePunctuation)
        corpus.tmp <- tm_map(corpus.tmp, stripWhitespace)
        corpus.tmp <- tm_map(corpus.tmp, removeNumbers)
        corpus.tmp <- tm_map(corpus.tmp, content_transformer(tolower))
        corpus.tmp <- tm_map(corpus.tmp, stemDocument)
        corpus.tmp <- tm_map(corpus.tmp, removeWords, stopwords("english"))
        corpus.tmp <- tm_map(corpus.tmp, removeWords, qn_stopwords)
        return(corpus.tmp)
}

doc_corpus <- VCorpus(DirSource(dir.name, encoding = "UTF-8"),
                      readerControl = list(language = "en"))
corpus.cleaned <- cleanCorpus(doc_corpus)

# TDM with all terms
tdm_cleaned <- TermDocumentMatrix(corpus.cleaned)

# trimmed TDM, discarding tf <= 2
tdm_cleaned_tf2 <- TermDocumentMatrix(corpus.cleaned,
                                      list(bounds = list(global = c(3, Inf))))
```

	terms	Non-/sparse entries
all terms (after preprocessing)	2223	17878/2583032
discarding tf = 1	1416	17071/1639649
discarding tf $\leq$ 2	1039	16317/1199313
discarding tf $\leq$ 3	846	15738/974082

Table 2.2: Total number of documents in QuantNet: 1170 Gestalten/1826 Quantlets; term sparsity: 98% – 99%

2.6.2 Sparsity results

The BVSM, GVSM(TT) and three LSA configurations with the dimension parameter k equal to 300, 171 (50% of the weight of all singular values) and 50 were considered, see Table 2.3. *Sparsity* and *density* are terms used to describe the percentage of cells in a database table (or a matrix) that are not populated and populated, respectively. The sum of the sparsity and density should equal 100%. Sparsity is the ratio of the number of zero entries to the total number of entries of a matrix. In general, the lower the sparsity, the better, see also "drawbacks of the BVSM" in Section 2.4.4.

Heat maps with dendrograms of the similarity matrices in the appropriate model configurations are displayed in Figures: 2.25, 2.26, 2.27, 2.28, 2.29. They allow an extensive visual interpretation and characterization of the inherent cluster structure of the included text documents. The method `heatmap.2` from the R package **gplots** was used for creating the heat maps (Warnes et al., 2016). This method simultaneously performs reordering of the matrix rows and/or columns according to the row and/or column means within the restrictions imposed by the dendrogram. Hence, an easier identification of "similarity clusters" within the matrix is provided. The color map on the left displays the meaning of the color keys: yellow values show the similarity values close to 1, red values those close to zero, see also Formula 2.2.

Two interesting effects can be stated. I) GVSM(TT) and LSA similarity matrices pronounce a higher concentration of "similarity clusters" around the diagonal than those in the BVSM, thereby indicating subsets of documents allowing good clusterization into one particular group.

II) LSA allows an adjusted sparsity reduction and similarity enhancement, respectively, by varying the k parameter. We can see the apparent relationship that lower k values imply clearer "similarity clusters" within the matrix, compare the Figures 2.27, 2.28 and 2.29.

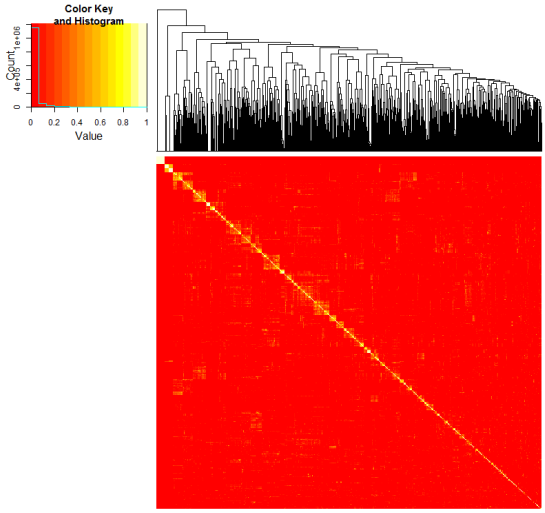


Figure 2.25: BVSM

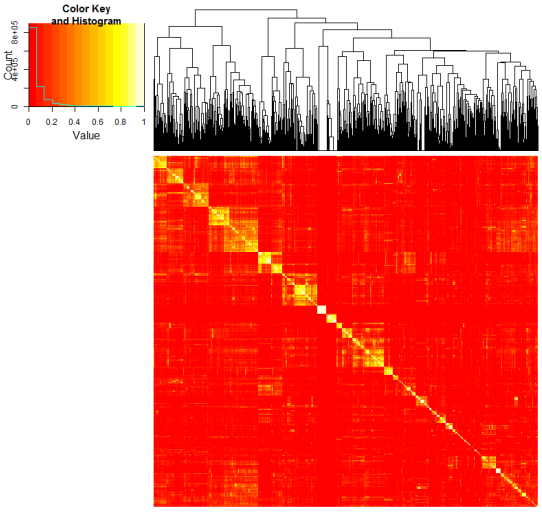


Figure 2.26: GVSM(TT)

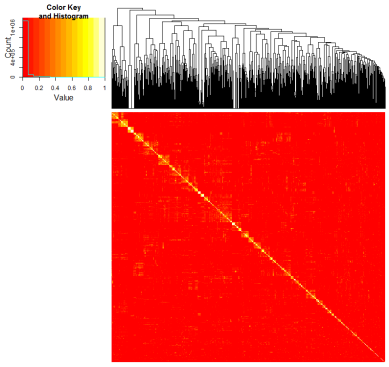


Figure 2.27: LSA:300

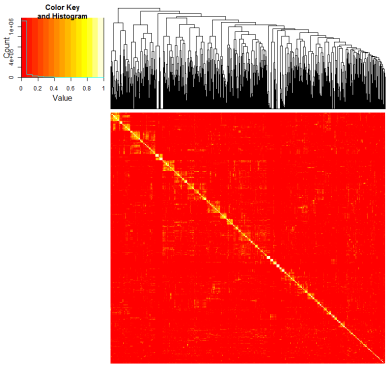


Figure 2.28: LSA:171(50%)

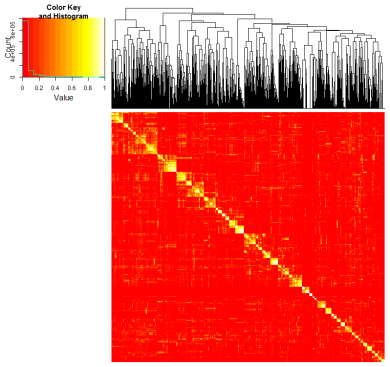


Figure 2.29: LSA:50

We can conclude that the more sophisticated models GVSM(TT) and LSA clearly outperform the BVSM, concerning both the TDM and similarity matrices. Given the pure numbers in Table 2.3, we observe that the LSA configurations reduce the TDM sparsity to the greatest extent. In the case of similarity matrices GVSM(TT) achieves the greatest sparsity reduction.

	BVSM	GVSM(TT)	LSA:300	LSA:171(50%)	LSA:50
TDM	0.99	0.65	0.51	0.51	0.47
M_S	0.65	0.07	0.35	0.36	0.35

Table 2.3: Model performance regarding the sparsity of the "term by document" matrix TDM and the similarity matrix M_S in the appropriate models (weighting scheme: tf-idf normalized)

2.6.3 3 Models, 3 Methods, 3 Measures

For evaluation and benchmark purpose we have introduced the so-called M^3 evaluation. All TM **models**, clustering **methods** and validation **measures** as presented in the previous sections are combined in a $3 \times 3 \times 3$ benchmark setup, hence the name M^3 evaluation. Every M stands for one of the dimensions: models, methods and measures.

- 3 models: BVSM, GVSM(TT) and LSA
- 3 clustering methods: k-means, k-medoids, HC
- 3 cluster validation measures: connectivity, Silhouette width, Dunn index

More precisely, the current experimental design should be named as $M^3_{3,3,3,250}$ (250 as the maximal cluster size to be evaluated). Later we will explain how it can be extended to $M^3_{d_1,d_2,d_3,max}$, with d_i encompassing more settings in the appropriate dimension.

Concerning the LSA, two configurations were taken: k equal to 171 (50% of the weight of all singular values) and 50. There is another implicit dimension in the experimental design, namely the number of possible clusters, let's call it i, which is captured on the x-axis in the plot matrix in Figure 2.30. We have decided to run the validation for the first $2, \ldots, 250$ i-values. Since our TDM has 1170 documents/columns, we regard the choice of 250 as the maximal cluster size as appropriate. On the one hand, 250 is more than enough for the practical needs. On the other hand, in the case of 250 clusters amongst 1170 objects one would obtain around 5 objects in one cluster at average. This is quite close to the extreme case, one object in one cluster, what is trivial and honored by the most validation measures with the "highest score". All things considered, our choice of the maximal cluster size was a good compromise between the practical needs, the theoretical limits and computational expense.

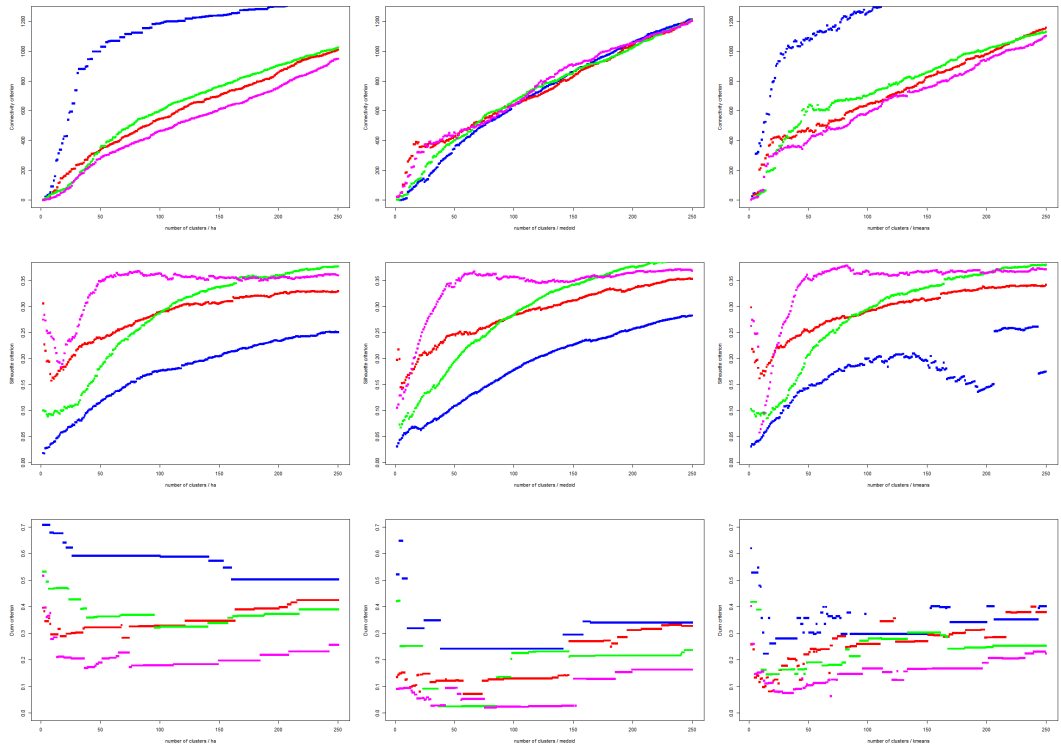


Figure 2.30: M^3 plot matrix. Rows: connectivity, Silhouette, Dunn. Columns: HC, k-medoids, k-means. Colors: BVSM, GVSM(TT), LSA, LSA50

Listing 2.2 demonstrates the main idea of the M^3 experimental design. For any given TDM the main function `clValid` is executed. Afterwards, the evaluation results for all considered TM models are aggregated with respect to any considered validation measure and clustering method, in our example, Silhouette and HC. Apparently, the experimental design can be extended in any dimension: more TM models, more clustering methods, more validation measures and, if necessary, more cluster sizes. The increasing calculation time of the overall experiment should be considered. A contemporary Intel Core i5 CPU needed one night to finish all calculations.

Listing 2.2: Cluster validation via the R package **clValid**

```
# load the R package
library(clValid)
# transpose the TDM in the LSA model
A = t(m_lsa_mat)
# run the main evaluation function
intern <- clValid(A, 2:250, clMethods=c("hierarchical","kmeans","pam"),
                  validation="internal")
# basic inspection methods
summary(intern)
plot(intern)
m_lsa = measures(intern)

# aggregate evaluation results for 4 different TM models; Silhouette / HC
x_l = 250
plot(2:x_l, m_b[3,,1], pch=15, ylim=c(0.01,0.7), col="blue",
     xlab="number of clusters / hc", ylab="Silhouette criterion")
lines(2:x_l, m_tt[3,,1], type = "p", pch=15, col="red")
lines(2:x_l, m_lsa[3,,1], type = "p", pch=15, col="green")
lines(2:x_l, m_lsa50[3,,1], type = "p", pch=15, col="magenta")
legend("topright", col= c("blue", "red", "green", "magenta"), pch=15,
       legend = c("BVSM","GVSM(TT)","LSA", "LSA50"), lty=3)
```

For any given M^3-combination (fixed measure, method and model) the shape of the function graph in the appropriate M^3 plot matrix cell (a particular row, column and color) can exhibit an individual behavior, see Figure 2.30. Characteristic for validation measures in our setup is the monotonous growth. In some cases there are some fluctuations and oscillations for lower i values. After an initial period of some i's all function graphs start to consolidate their growth trend. Remarkable is the unstable and noisy behavior of the k-means method, in particular in the BVSM. Another interesting observation is the combination Silhouette and LSA50. First the graph has a strong oscillation with a decreasing trend, then a relatively steep ascent and finally, after around a quarter of the interval length of i-values, the graph shows a stable sideways movement.

Measure	Model	Method
Connectivity	LSA50	HC
Silhouette	LSA50	HC
Dunn	BVSM/LSA	HC

Table 2.4: M^3 evaluation results

The results of our M^3 evaluation are summarized in Table 2.4. The most important observations and conclusions are:

- HC better or comparable to other methods under all measures and in all models
- LSA50 superior with respect to the connectivity and Silhouette measures
- BVSM/LSA slightly better than LSA50 with respect to the Dunn measure, but still comparable (small range of values in all models)
- Conclusion: LSA/LSA50 and HC is the optimal model/method combination under M^3 evaluation

2.6.4 LSA anatomy

Since the SVD truncation as performed in Section 2.4.4 results in the following decomposition:

$$D = D_k + D_{err} \tag{2.11}$$

and

$$D_k = U_k \Sigma_k V_k^\top, \tag{2.12}$$

the question arises how these six matrices, namely $D, D_k, D_{err}, U_k, \Sigma_k$ and $V_k^\top$, look like?

SVD decomposition

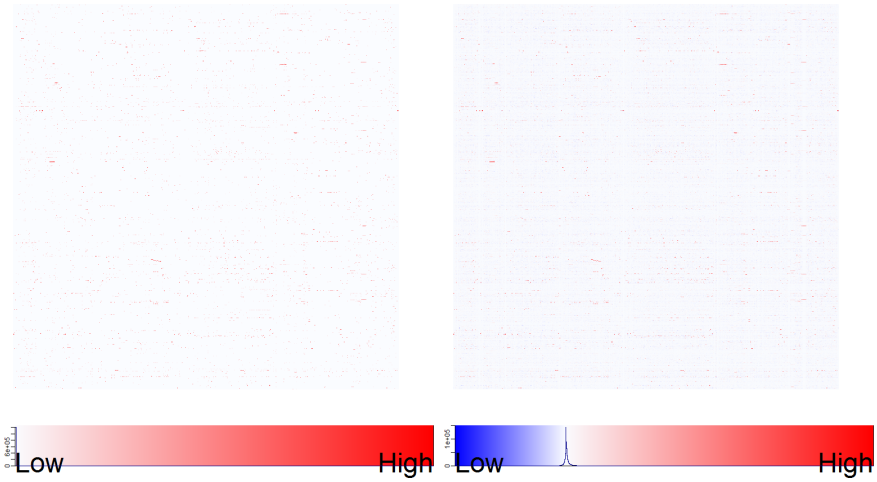


Figure 2.31: Heat maps with color key of D and D_k (from left to right)

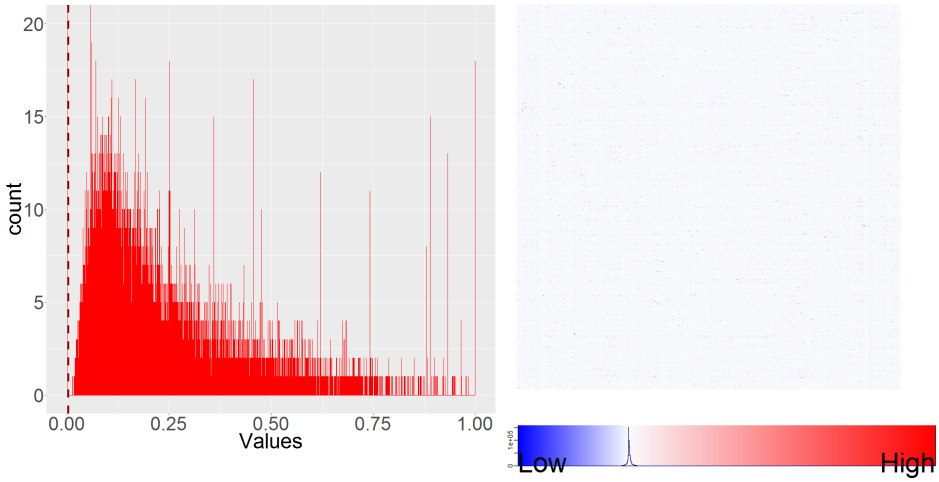


Figure 2.32: Histogram of the matrix values in D (to the left); heat map with color key of the error matrix D_{err} (to the right)

All results, in particular all plots and figures, from this subsection can be examined and reproduced by the corresponding Quantlets, available under Q3D3LSA. The reader can also "just browse" through the GitHub repository and study the plots in a higher resolution, in particular the high dimensional matrix representations. The most important incorporated R packages are **lsa** (Wild, 2015), **gplots** (Warnes et al., 2016) and **ggplot2** (Wickham, 2009). In

the beginning of every Quantlet the LSA space is created from the term document matrix TDM of the Quantlets, which was created as described in Section 2.6.1.

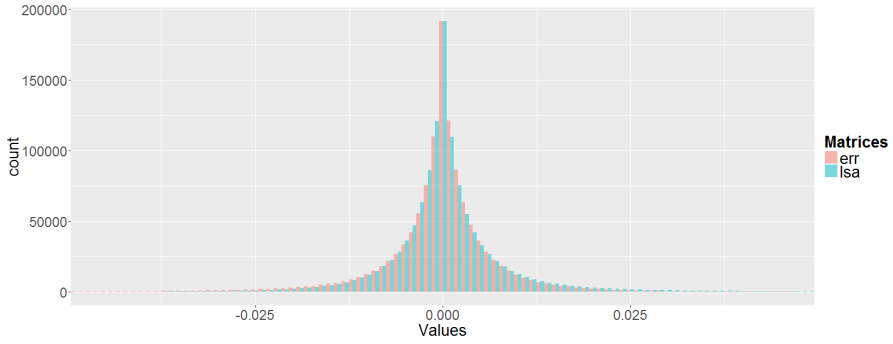


Figure 2.33: Histograms of D_k and D_{err}

In Figures 2.31, 2.32 and 2.33 we first see the heat maps of D, D_k and D_{err} and then the histograms of the distribution of the corresponding matrix values (LSA_heatmaps_sum, LSA_basics, LSA_basics_hist_box). D is our original TDM from the BVSM, D_k is the corresponding truncated and re-embedded TDM from the LSA model (reduced to the first k dimensions) and D_{err} is the approximation error matrix of the SVD truncation.

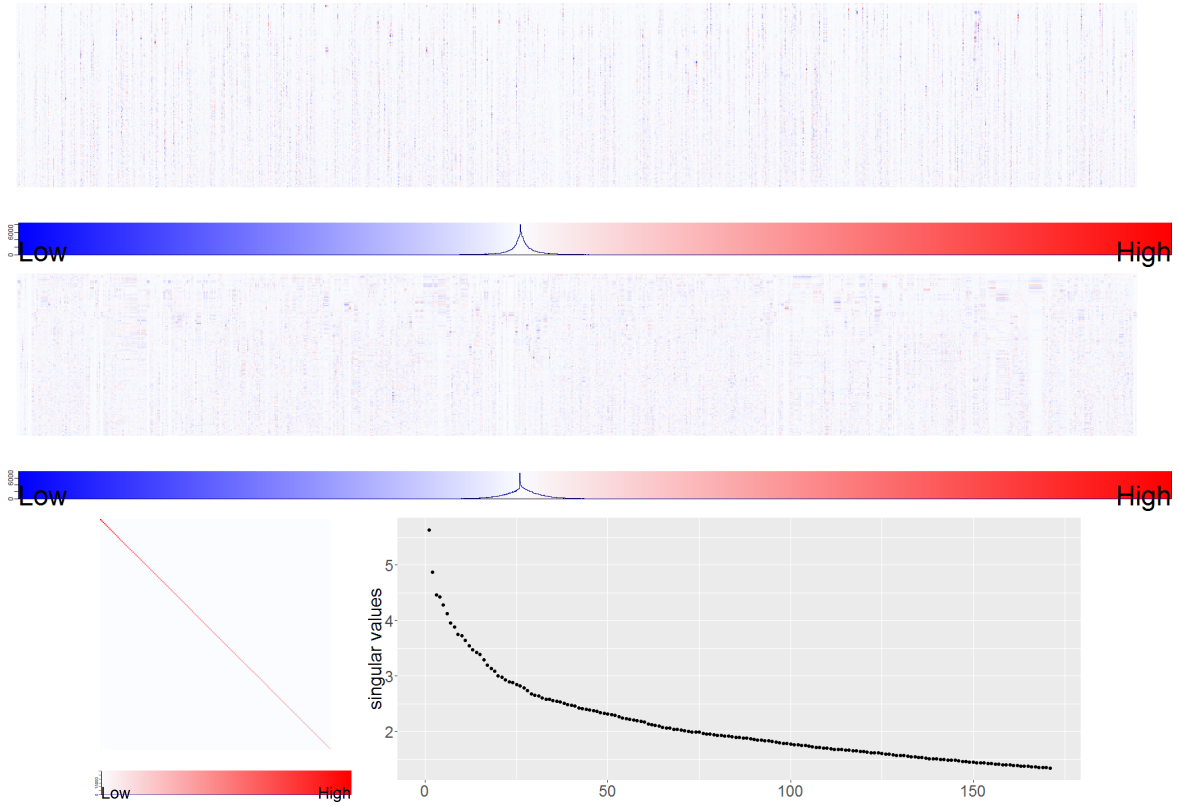


Figure 2.34: Heat maps of $U_k^\top$, $V_k^\top$, Σ_k (from top to bottom); plot showing the highest singular values having a total sum of 50%

The SVD factor matrices U_k, Σ_k and $V_k^\top$ from Formula 2.12, their histograms and boxplots are displayed in Figures 2.34, 2.35 and 2.36. The heat maps are produced via LSA_heatmaps_factors. For reasons of space, the matrices U_k and $V_k^\top$ are both displayed in a "horizontal way".

The examined matrices reveal the following characteristics. All of them are sparse, what becomes evident when looking at their color maps. The red color means positive matrix values close to 1, white means values equal to zero and blue displays negative values covering a subset of the interval $[-1, 0]$. While the matrix D has only non-negative values (what is clear from the definition in Formula 2.1), the matrices D_k and D_{err} include also negative ones. Both of them are concentrated on a relatively small interval $[-0.05, 0.05]$. D_k exhibits a higher shift towards positive values, whereas D_{err} pronounces a higher shift to the negative ones. D_k has a smaller sparsity than D due to its distribution properties, compare also Table 2.3. D_{err} behaves more like a typical statistical error term ϵ.

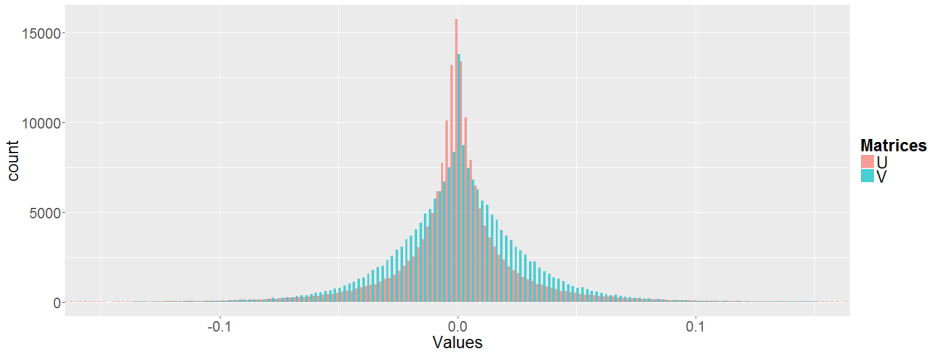


Figure 2.35: Histograms of $U_k^\top$, $V_k^\top$

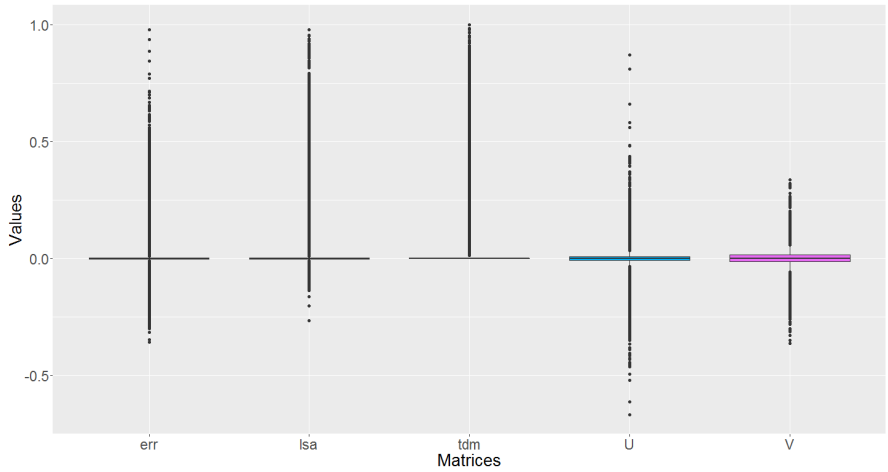


Figure 2.36: Boxplots of D_{err}, D_k, D, $U_k^\top$, $V_k^\top$

The distribution of the matrix values in U_k ($U_k^\top$ resp.), whose columns (rows resp.) represent the "semantic components", can be appreciated by means of the heat map and histogram (LSA_heatmaps_factors, LSA_basics_hist_box). The distribution in Figure 2.35 indicates some kind of a "symmetric Pareto distribution" behavior. Upper rows (i.e. rows with larger singular values) of $U_k^\top$ in Figure 2.34 show higher concentration of non-zero values on several terms (i.e. columns). Downwards, the distribution of term frequencies (columns values) is getting more diffuse and chaotic. A possible intuition could be that the lower "semantic components" (rows) with lower singular values contribute consistently to all terms, but the contribution impact is smaller. A lot of term values of the upper row coordinates of the matrix are close to zero, for a smaller number of terms there is a distinct change of colors, i.e. of term weights. Thus, the upper "semantic components" (with larger singular values) influence some terms especially strong through all documents.

Concerning the matrix $V_k^\top$, we observe in the heat map in Figure 2.34 several subsets of neighboured columns which look very similar, i.e. having nearly the same values (colors) in the row coordinates. Every column $v_{k,j}^\top$ in $V_k^\top$ contains the coefficients for any given document d_j^{LSA} (representing a particular Quantlet) in the truncated LSA space:

$$d_j^{LSA} = U_k \Sigma_k v_{k,j}^\top. \tag{2.13}$$

Hence, we can conclude that it is possible to observe the similarity of a group of documents on the basis of the similarity of their coefficient vectors $v_{k,j}^\top$. This is justified by the fact that the linear mapping $x \mapsto U_k \Sigma_k x$ is continuous in x (with respect to any norm). This is in particular important for big data sets as the dimension of $V_k^\top$ is $k \times n$, with n as number of documents. Since the dimension of the original feature space is m ($dim(U_k) = m \times k$), one should consider to perform document clustering not in m but only in k dimensions of the latent LSA space. The number of terms m is fixed, whereas k can be controlled and reduced via the SVD process. The price of this is the approximation error D_{err}.

The shape of the distribution of the matrix values in $V_k^\top$ is comparable to that of U_k, see Figure 2.35. In the case that the assumption of a symmetric Pareto distribution should hold, the distribution of $V_k^\top$ would have a smaller shape parameter α. This would also explain the higher concentration of the distribution of U_k on a smaller interval around zero, apart from some outliers. Due to the fact that $V_k^\top$ and U_k are truncations from orthogonal matrices, no observations outside of the $[-1, 1]$ interval can be made.

With respect, finally, to the diagonal and quadratic matrix Σ_k, everything is self-explaining when looking at its heat map and plot in Figure 2.34. The parameter k was chosen in such a way that the biggest singular values with a subtotal of 50% were maintained, which resulted in $k = 171$. All steps can be reproduced by LSA_basics.

Semantic kernel

The "semantic kernel" $U_k U_k^\top = U I_k U^\top$ as introduced in Section 2.4.4 can be interpreted as correlation of terms in the lower k-dimensional semantic space. LSA_kernel allows random samples of the full semantic kernel and further experiments and visualizations as shown in the following figures.

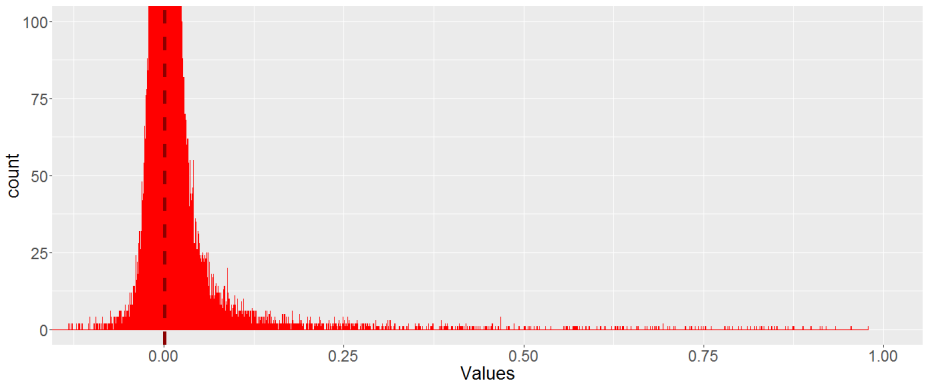


Figure 2.37: Histogram of the full semantic kernel $U_k U_k^\top$

Figure 2.38 shows a randomly chosen 30×30 sub matrix of the kernel. In opposite to the BVSM, where the terms are orthogonal, LSA allows for establishing a correlation structure between different words/terms. The word stem pairs "optionpric/price", "dividend/binomi", "put/dividend", "put/optionpric", "factor/score" etc. illustrate the statistical co-occurrence information extracted by means of the SVD.

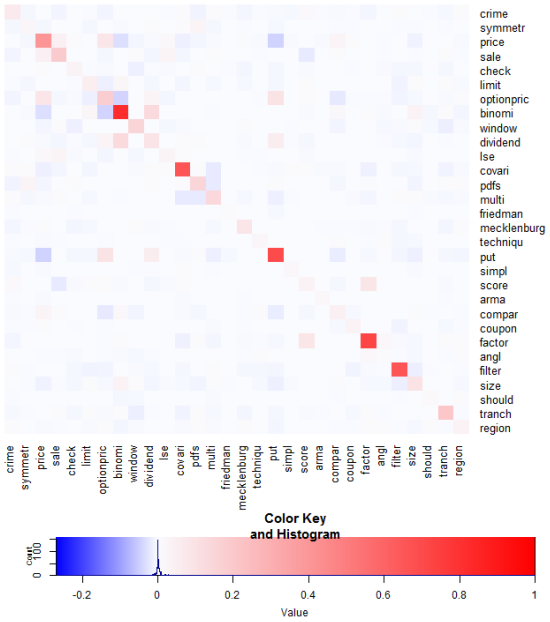


Figure 2.38: Heat map of semantic kernel $U_k U_k^\top$, random subset 30×30

Figures 2.37 and 2.39 visualize the full semantic kernel. The kernel $U_k U_k^\top$ is a symmetric 1039×1039 matrix, every entry of which represents the correlation between two given terms (dimensions of the matrix). The histogram in Figure 2.37 shows the distribution of the correlation in a relevant range, indicating that the semantic kernel has a distinct shift towards positive correlations.

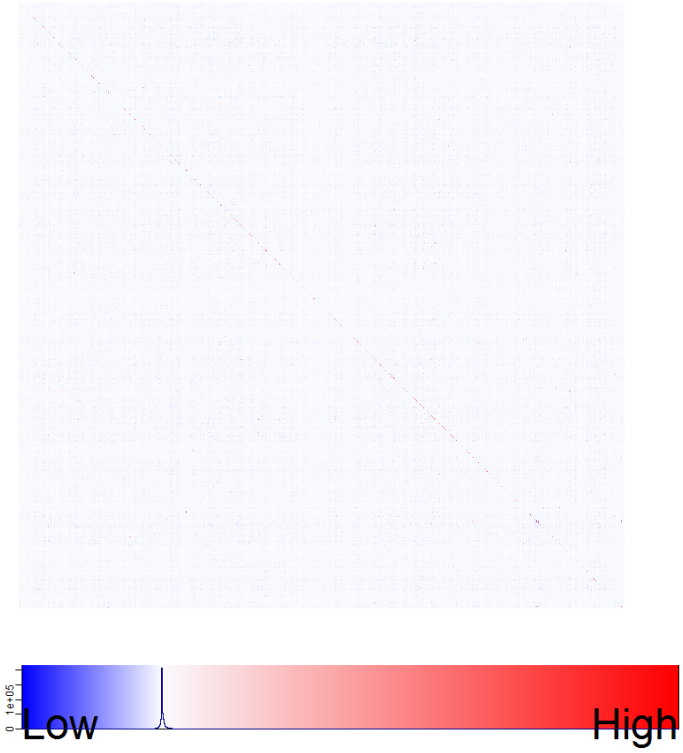


Figure 2.39: Heat map of the full semantic kernel $U_k U_k^\top$

Semantic space versus HC

A fundamental question that arises is how the "semantic components", columns of the matrix U_k or left-singular vectors in mathematical terms, can be interpreted? Since the singular vectors (columns of U and V) are only unique up to scalar multiples of modulus one, in the case of real matrices +/- 1, the appropriate choice of the sign (or "rotation") should be considered. LSA_PC_rotation determines the proper sign (rotation) of the PC's (semantic space Principal Components) and extracts the top words of each PC for the given LSA space.

No determination of PC's rotation:
PC1 (5.6): visual return option call distribut
PC2 (4.9): call option blackschol put price
PC3 (4.5): dsfm fpca dsfmbsyc dsfmfpcaic cluster
PC4 (4.4): dsfm copula distribut densiti gumbel
PC5 (4.3): return visual portfolio timeseri dax
PC6 (4.1): regress kernel nonparametr linear estim
PC7 (4.0): copula regress gumbel nonparametr var
PC8 (3.9): copula visual gumbel scatterplot clayton

Auto determination of PC's rotation:
PC1 (5.6): visual return option call distribut
PC2 (4.9): visual densiti distribut copula normal
PC3 (4.5): dsfm fpca dsfmbsyc dsfmfpcaic cluster
PC4 (4.4): cluster analysi pca dendrogram principalcompon
PC5 (4.3): copula normal distribut gumbel call
PC6 (4.1): return copula cluster portfolio gumbel
PC7 (4.0): copula regress gumbel nonparametr var
PC8 (3.9): requir stflosst stfstab distribut approxim

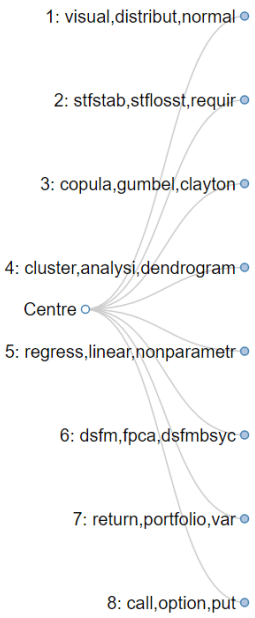


Figure 2.40: Topics of the first 8 "semantic components" (LSA on the left) versus cluster labels of the dendrogram (HC on the right)

In more detail, the positive or negative sign is chosen based on the weights of the positive and negative part of the PC (column of U_k). More precisely, PC^+ and PC^- are compared and the sign is changed (the PC is multiplied by -1), if $\|PC^-\|_1 > \|PC^+\|_1$. As a reminder: $f^+(x) = \max(f(x), 0)$, $f^-(x) = -\min(f(x), 0)$, for a real-valued function or vector f. Finally, the top words with the highest (positive) weights are taken as "prototypes" for the PC's topics, thereby allowing the determination of possible labels for the "semantic components".

Figure 2.40 shows on the left side two possible outputs of LSA_PC_rotation: one PC representation without sign correction (no rotation), and the other with correction (auto determination of rotation). According to Table 2.5, in five of eight cases the sign correction was performed, determining other topic labels for the corresponding PC's. The cases with sign correction are marked with a star.

For comparison reasons, the two PC topic alternatives were contrasted against the labels of the dendrogram clusters[11], see also Figure 2.40. The coincidence of the topic terms in the "semantic components" and in the labels of the dendrogram clusters are summarized in Table 2.5. We can observe the interesting effect that there is a perceptible correspondence between the PC topics and HC cluster labels. A correspondence was recorded every time there were

[11] `https://github.com/Quantlet/D3Genesis` ; "V. QuantNet full - Hierarchical cluster analysis (3 levels) with Topics (Aug 30, 2015); Expandable Tree"

at least two common terms both in the PC topics and HC labels. In more than half of the cases even an unambiguous matching between PC and HC topics/labels is possible. In the case of PC's without rotation correspondences towards six of eight HC cluster labels are possible ($\{1, 3, 5, 6, 7, 8\}$). In the case of auto-rotated PC's correspondences towards all eight HC clusters can be recorded. Therefore it can be concluded that from the semantic point of view "semantic components" (PC's) and cluster labels (HC) provide good approximations to each other.

PC nr (no rotation)	cluster nr (HC)	PC nr (auto rotation)	cluster nr (HC)
1	$1, 8$	1	$1, 8$
2	8	$2^\star$	1
3	6	3	6
4	3	$4^\star$	4
5	7	$5^\star$	$3, 1$
6	5	$6^\star$	$7, 3$
7	$3, 5$	7	$3, 5$
8	3	$8^\star$	2

Table 2.5: Coincidence table of PC numbers versus HC cluster numbers

2.7 Application

The current implementation of the self-developed visualization framework for knowledge discovery in QuantNet is displayed in Figure 2.41. The so-called D3 Visu application is available as web page at `http://quantlet.de`. Driven by the Q3-D3-LSA technology, which is the combination of our research findings, the integrated search engine facilitates easier discovery of shared validated knowledge and collaborative reproducible research (CRR). While the D3 based application provides an interactive front end of IR, document clustering and visualization elements, one can rely on the robust data storage infrastructure of GitHub in the background, comprising the distinct abilities of version control (VC) and source code management (SCM). A start page screenshot of the Quantlet GitHub organization is given in Figure A.1.

The GitHub platform, having more than 14 million users and more than 35 million repositories, is currently the largest host of source code in the world. It provides access control, task management and collaboration features for all project types. Thanks to the Style Guide[12], Yamldebugger R package[13] and introductory Quantlets yamldebugger_intro, the Quantlet organization members[14] have all necessary tools for a fast, transparent and iterative code development and documentation process. Once a member or outside collaborator has contributed valid Quantlets, the TM pipeline retrieves the meta information of Quantlets via the GitHub-R-API and distills them to human-readable and applicable information by means of the Q3-D3-LSA technology.

Quantlets, which have been processed in that manner, are finally extracted into the D3 Visu application layer, called QuantNetXploRer. Figure 2.41 demonstrates a typical application. The hits of the entered search query, in this case "big data", are displayed both in textual form as in graphical form. Quantlets (represented by nodes) containing the expression "big data" are highlighted in red color. The application screen is divided into the central main visualization

[12] `https://github.com/Quantlet/Styleguide-and-FAQ`
[13] `https://github.com/Quantlet/yamldebugger`
[14] `https://github.com/orgs/QuantLet/people`

("orbit clustering" scheme), and auxiliary components like buttons, tool tips and legends. The upper control panel allows the choice of different clustering schemes, D3 layouts, color palettes and allows the configuration of the dynamic and draggable legends. Two legends allow to filter the nodes by programming languages or books and projects. Other two legends display the cluster topics or GitHub repositories of the visualized Quantlets. All relevant auxiliary components are draggable, can be deactivated and are responsive, what means that the action performed in one element is reflected in all other visualization components. For instance, if the user filters the nodes by the programming language R, the contents of the main D3 Visu, the cluster topics legend and the GitHub repositories legend are updated. The statistic of the remaining language combinations in the programming languages legend is recalculated, too. All updates of the main D3 Visu are realized via dynamic transition effects.

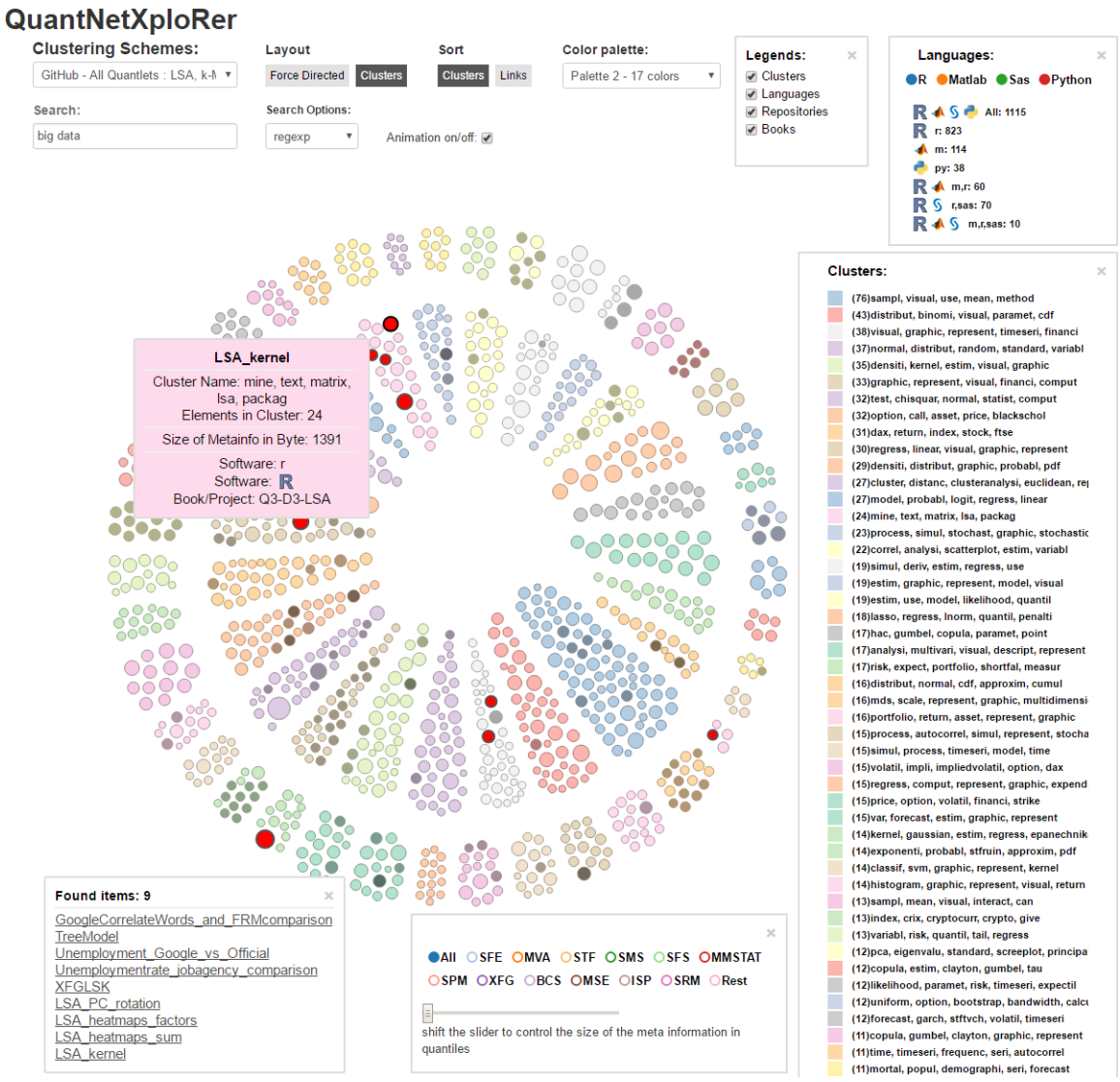


Figure 2.41: Front end view: all Quantlets in QuantNetXploRer, search term "big data"

The Q3-D3-LSA engine of the QuantNetXploRer has many other characteristics and features which are best explored by "learning by doing":

Build Quantlets better, together, now (QuantNet @ GitHub).

2.8 Outlook

The benchmarks in Section 2.6 have shown that different GVSM configurations allow adapted similarity based document clustering. Concerning sparsity and higher concentration of "similarity clusters" (as shown in Section 2.6.2) both the GVSM(TT) and LSA configurations clearly outperform the classical BVSM. Incorporating term-term correlations and semantics, GVSM(TT) and LSA provide considerable sparsity reduction, thereby achieving higher clustering performance. The main advantage of LSA is the flexible dimension reduction property which is controlled by the truncation parameter k within the SVD process. Additionally, the M^3 evaluation identifies the LSA/LSA50 and HC as the optimal model/method combination. The benefits of the dimension reduction effect with smaller k values can be also observed in the M^3 plot matrix (see Figure 2.30).

First benchmark results in Section 2.4.4 show that the LSA model seems to be applicable for big data and has a modest time complexity. Thus, samples of 100.000 GitHub organizations could be processed within an hour. Potential bottlenecks are the GitHub API extraction process or the calculation of big distance matrices for some clustering methods. Both issues could be tackled by massive parallelization and are beyond the actual subject "TM models".

2.8.1 GitHub Mining Infrastructure in R

Our TM pipeline together with the GitHub-R-API implementation relies on several sophisticated R packages like **tm**, **lsa**, **svd**, **cluster**, **yaml**, **jsonlite** and some more. An essential element is the R package **github** "R Bindings for the Github v3 API" (Scheidegger, 2016). Taken as a whole, we have a powerful "GitHub Mining infrastructure in R" which allows to incorporate any GitHub organization with its content for further analysis and possible data mining thanks to the official GitHub API[15]. Currently, there are more than one million organizations on GitHub, among them Google, Facebook, Twitter, Yahoo, CRAN, RStudio, D3, Plotly and many more. The BitQuery[16] is a current project which allows to mine some popular GitHub organizations containing several ten thousand repositories.

Listing 2.3 shows how the content of any publicly available repository on GitHub can be retrieved within R. The first parameter `owner` can be substituted by any organization or user name. Basically, the operating and mining scope of QuantNet can be extended to any subset of GitHub. One challenge is to implement the appropriate parsers for the specific repository structures and contents of new organizations. The other is to adjust and calibrate the TM models to the new kind of information. Actually, QuantNet has already several parsers implemented. In addition to the Quantlet organization, the repository "Introduction to Statistics with Python"[17] (Haslwanter, 2016) is also incorporated via the Q3-D3-LSA engine.

Listing 2.3: GitHub API method **Get contents** returns the contents of a file or directory in any repository on GitHub which is publicly available

```
get.repository.path <- function(owner, repo, path,
                        ..., ctx = get.github.context())

# Browser as function for
# https://github.com/thomas-haslwanter/statsintro_python
QBrowser_2Dir_Offset = function(gh_user = "thomas-haslwanter",
                        reponame = "statsintro_python",
                        path_offset = "ISP/Code_Quantlets", showSummary = TRUE)
    ### Start
    rep_c = get.repository.path(gh_user, reponame, path_offset, ctx = ctx)
```

[15] https://developer.github.com/v3/
[16] https://github.com/bemined/BitQuery
[17] https://github.com/thomas-haslwanter/statsintro_python

2.8.2 Future Developments

In the near future, I am going to publish 3 R packages under the overall heading "GitHub API based QuantNet Mining infrastructure in R". At this stage it seems reasonable to organize this R infrastructure in the following packages:

- **rgithubQ**: an extension of the R package **github**, first of all, enabling file operations like *Create a file*, *Update a file* and providing a series of low level API helping functions[18].
- **tmPipelineQ**: comprising the parser layer, TM models layer, clustering layer and D3 export layer. This is the main component of the Q3-D3-LSA engine.
- **mdGeneratorQ**: GitHub Markdown generator, a special extended parser runs trough the QuantNet repository structure, extracts resources like meta information, source code, pictures etc., reformats, integrates and exports them via the GitHub API into a single Markdown file for every Quantlet, see e.g. Figure 2.24.

The prototypes of the aforementioned 3 packages are already in operational and working state and are continuously tested and improved. The **tmPipelineQ** and **mdGeneratorQ** prototypes operate independently from each other. Both components require the **rgithubQ** functionality. The final design and structure of the "GitHub API based QuantNet Mining infrastructure in R" packages is subject of current research and will be presented in Borke and Härdle (2017).

Furthermore, more TM models, clustering methods and validation measures could be considered and studied for performance validation: from M^3 to $M^3_{d_1,d_2,d_3,max}$, see Section 2.6.3. Optimization of the automatically generated cluster labels for easier human readability and implementation of new "upgrades" into the D3 Visu could contribute to a better usability of the Q3-D3-LSA technology.

[18] `https://github.com/cscheid/rgithub/blob/master/todo.org`

3 GitHub API based QuantNet Mining infrastructure in R

3.1 Introduction to GitHub Mining

This research is the technological and scientific basis of the project "GitHub API based QuantNet Mining infrastructure in R" as introduced in Borke and Härdle (2016). Its structure and objectives are described by the diagram in Figure 3.1 and in the following text. The research starts with the *Parser 1* node of the TM Pipeline (see Figure 3.1) and goes along the path till the end point at the *Smart Clusterization* node. Alternatively, the pipeline could start with other parsers, depending on the data source. For instance, *Parsers* 2 or 3 could be used for processing the special repository structure of papers or external books, respectively.

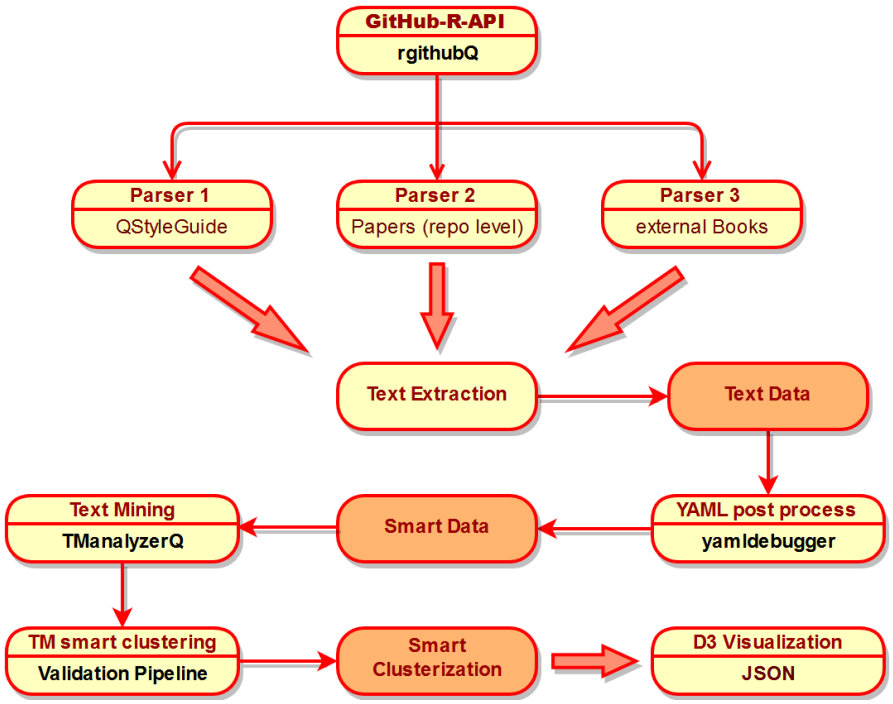


Figure 3.1: TM Pipeline of the "GitHub API based QuantNet Mining infrastructure in R"

An integral part of the overall project is **GitHub** – a Git repository hosting service founded in 2008, which not only provides its users with a web-based graphical interface of the well-known version control system, but also allows such high-level features as access control and collaborative work (Loeliger, 2009). The first essential step of the large scheme is to adjust the application program interface (API) of GitHub to the R software environment and to implement different parsers for data extraction from various repositories of programming projects, see the **rgithubQ** package (Scheidegger and Borke, 2017). Big Data obtained at this stage is thus constituted by a large corpus of text documents, each of which corresponds to the meta information of a particular program code, be it in a repository, a folder or under another GitHub storage resource.

QuantNet was originally designed as a platform to freely exchange empirical as well as quantitative-theoretical methods for statistical and economical programming, called Quantlets, or in abbreviated form "QLs". It supports the deployment of computer codes written in R, Matlab, SAS and Python. Because of the open structure other languages can be easily added.

The first objective is to implement initial processing of the massive text data obtained from the QuantNet's GitHub organization, available at `https://github.com/Quantlet`. In the case of QuantNet, the task of extracting smart data out of the raw text collection is completed by means of the **rgithubQ** and the **yamldebugger** packages (Borke, 2017d), using the YAML (`http://yaml.org/`) encoded metadata of the QLs, see Sections 3.3 and 3.4. Being a human-readable data serialization language, YAML is commonly used for configuration files, but could be used in many applications where data is being stored (e.g. debugging output) or transmitted (e.g. document headers). Thus derived smart data pass through a compound chain of processing layers, TM and Smart Clustering layer. The TM layer is implemented in form of the **TManalyzerQ** package (Borke, 2017c), see Section 3.6. The Smart Clustering layer is calibrated via the **Validation Pipeline** (*Vali-PP*) as described in Section 3.7.

The second objective is to calibrate the IR performance and effectiveness in different TM models. Section 3.5 shows how the search queries obtained from Web Metrics via the **RGoogleAnalytics** package (Pearmain et al., 2014) can be exploited for this purpose. Further, different clustering and validation methods within the R software environment are examined in order to determine the optimal combination of the data configuration, vector space model, clustering method and clustering criteria settings. The *Vali-PP* evaluates thereby the resulting partition and eventually finds a reasonable number of clusters, see Section 3.7.1 for more details. While Section 3.2 provides the related work of this research, Section 3.8 presents an overview of the findings.

In a summary, the GitHub-R-API based and **rgithubQ** driven TM pipeline (including three parser types as displayed in Figure 3.1) retrieves the YAML encoded meta information of Quantlets via the **yamldebugger** package, then the LSA model is applied, clusters and labels are generated (by use of the **TManalyzerQ** package and **Validation Pipeline**) and the processed data is transferred via JSON into the D3 application, which is the visualization layer of the QuantNetXploRer.

3.2 Related Work

D3.js (or just D3 for Data-Driven Documents) is a JavaScript library for producing dynamic, interactive data visualizations in web browsers. The QuantNetXploRer is a good example of D3 in power. More information about the D3 architecture, its various designs and the D3-based QuantNetXploRer can be found in Bostock et al. (2011) and Borke and Härdle (2016). The repository `https://github.com/Quantlet/D3Genesis` contains detailed information about the development of the main D3 components for the QuantNet visualization together with live examples on GitHub pages.

One of studies presenting the effectiveness of LSA was performed by Feinerer and Wild (2007). They applied LSA based algorithms for the automated processing of transcripts of interviews. Compared to marketing expert judgments, the machine results showed very high levels of reliability and validity in automatic text analysis. Moreover, the LSA approach proved useful not only in avoiding human inherent subjectivity, but also in reducing high costs of human judgment.

Wild and Stahl (2007) described the **lsa** R package (Wild, 2015) and illustrated its proper use through examples from the areas of automated essay scoring and knowledge representation. Feinerer et al. (2008) presented the **tm** package (Feinerer and Hornik, 2015) which provides a framework for text mining applications within R, encompassing techniques for count-based

analysis methods, text clustering, text classification and string kernels. The new package **TManalyzerQ** (Borke, 2017c) combines and extends the functionality of both packages, facilitating IR tools in 3 text mining models: BVSM, GVSM(TT) and LSA. As presented in Cristianini et al. (2002) and Borke and Härdle (2016), all three TM models are special representations of the generalized vector space model (GVSM).

Brock et al. (2008), Desgraupes (2013) and Charrad et al. (2014) provided a good overview about the existing clustering validity indices. Additionally, there are accompanying R packages for their application allowing cluster validation and determining the relevant number of clusters in a data set: **clValid**, **clustCrit**, **NbClust**.

Gousios and Spinellis (2012) performed a deep analysis on the architecture of GitHub data and provided the overall schema of GitHub's data and API, discussing ways to overcome its limitations. Their paper also presents GHTorrent, an effort to create a scalable, queriable, offline mirror of data offered through the GitHub REST API, providing users with a possibility to analyze the development of their projects, and researchers with efficient tool to gather and analyze GitHub's event-stream data. Based on GHTorrent data, Kalliamvakou et al. (2014) analyzed the quality and properties of the data available from GitHub and discussed both positive and negative points of using and mining GitHub.

Scheidegger and his co-authors (North et al., 2015) introduced the design and implementation of the RCloud[1], an Integrated Exploratory Analysis, Visualization, and Deployment on the Web. Being an environment for collaboratively creating and sharing data analysis scripts, RCloud encompasses analysis code in R, HTML5, Markdown, Python, and others. Amongst other features, RCloud provides an environment, in which R packages can create rich HTML content, using, for example, D3 and dc.js, and a transparent, integrated version control system. RCloud's implementation[2] of the versioning mechanism is built on top of GitHub's gists[3]. The **github** package (Scheidegger, 2016) supports, amongst many other features, creating, modifying and administrating of GitHub's gists via the GitHub API.

The new package **rgithubQ** (Scheidegger and Borke, 2017), which extends the functionality of the **github** package, allows a GitHub wide search for code and repositories using the GitHub Search API. Performing similar as Google, it is designed to find results that best meet the personal needs and which are ranked by best match, as indicated by the score field for each item returned. The novel package introduces the QuantNet@GitHub statistics and some lightweight parsers. The current "QuantNet@GitHub" statistics are retrieved in real time as displayed in Table 3.1 (on February 28, 2017). First we see there the total number of all QLs on GitHub, then all QLs in the `https://github.com/Quantlet` organization. In the third place, all Style Guide compliant and validated QLs from the QuantNetXploRer visualization are displayed (`http://quantlet.de/`). Additionally, an optional character vector as an argument for the desired author/editor list is supported.

	total number
full_gh	2799
quantlet_gh_org	1495
QuantNetXploRer	1239
bykovskaya.as.editor	203
borke.as.editor	149

Table 3.1: QuantNet@GitHub statistics via the *qnet.stats* function from the **rgithubQ** package

[1] `https://github.com/att/rcloud`
[2] `http://rcloud.social/gallery/index.html`
[3] `https://help.github.com/articles/about-gists/`

3.3 QuantNet Search Code in a nutshell

The GitHub Search API[4] can be used for software mining of QuantNet and other GitHub organizations. Just like searching on Google, people want to see a few pages of search results so that one can find the item that best meets the personal needs. To satisfy that need, the GitHub Search API provides up to 1.000 results for each search. GitHub's results are sorted by best match, as indicated by the score field for each item returned.

```
library(rgithubQ)
library(yamldebugger)
# GitHub's user authorization
ctx = interactive.login("client_id", "client_secret")

q_search = 'Quantlet Published Description Keywords Author filename:"metainfo.txt"'

spec_search_term = "yaml user:Quantlet user:lborke user:b2net"
sr = search.code(paste(spec_search_term, q_search), per_page = 20)

spec_search_term = "black scholes user:Quantlet"
sr = search.code(paste(spec_search_term, q_search), per_page = 10)

sr$content$total_count

q_top = yaml.parser.light(sr, print_item = FALSE)

(q_names  = sapply( q_top, function(yaml_meta){ yaml.getQField(yaml_meta, "q")} ))
(q_author = sapply( q_top, function(yaml_meta){ yaml.getQField(yaml_meta, "a")} ))
(q_scores = sapply( sr$content$items, function(item){ item$score} ))
(q_repos  = sapply( sr$content$items, function(item){ item$repository$full_name }))

( name_path_scores = data.frame(qlet.name = q_names, qlet.repo.path = q_repos,
                                search.score = round(q_scores, 2)) )

a_splitted = unlist(str_split(q_author, ", "))
( tab_sorted = sort(table(a_splitted), decreasing = T)[1:10] )

( q_s = qnet.stats(spec_editor = c("bykovskaya", "borke")) )
```

Listing 3.1: QuantNet Search Code via the **rgithubQ** package

Listing 3.1 produces the results for Tables 3.1, 3.2, 3.3 and 3.4. After loading the package **rgithubQ** and "Basic Authentication"[5] the *Search code* functionality of the GitHub Search API is fully available. Two search queries are performed by the function `search.code`. The first search query retrieves all YAML meta infos according to the Style Guide and containing the term "yaml" from 3 users/organizations: Quantlet, lborke, b2net (see Table 3.2). Due to the parameter `per_page = 20` only the top twenty matches (sorted by the score value) are retrieved. These results are presented in Table 3.2. The second query specifies all meta infos in the organization `https://github.com/Quantlet`, which share the term "black scholes", see Table 3.3. As `per_page` was set to 10, only the top ten results are collected. Additionally, the variable `sr$content$total_count` provides the total count of all matches, 98 QLs in this case. The top ten authors (concerning the number of contributions) of the QLs retrieved by the second query are presented in Table 3.4. For that purpose `per_page` was set to 100 in order to capture all relevant QLs. Via the function `qnet.stats` the current "QuantNet@GitHub" statistics are retrieved in real time as displayed in Table 3.1.

[4] `https://developer.github.com/v3/search/`
[5] `https://developer.github.com/v3/search/#rate-limit`

	qlet.name	qlet.repo.path	score
1	yaml_run	lborke/yamldebugger_intro	12.05
2	yaml_keyword_finder	lborke/yamldebugger_intro	11.97
3	yaml_start	lborke/yamldebugger_intro	11.71
4	YAMLcentroids	b2net/Clustering_Validation_Pipeline	11.70
5	YAMLcleanmerge	b2net/Clustering_Validation_Pipeline	11.70
6	YAMLnumbchars	b2net/Clustering_Validation_Pipeline	11.70
7	lsa_heatmaperr	b2net/Clustering_Validation_Pipeline	11.70
8	lsa_heatmapsvd	b2net/Clustering_Validation_Pipeline	11.70
9	yaml_wordcloud	lborke/yamldebugger_intro	11.67
10	yaml_keyword_frequency	lborke/yamldebugger_intro	11.65
11	lsa_determineSign	b2net/Clustering_Validation_Pipeline	11.50
12	yaml_TDM_CorrPlot	lborke/yamldebugger_intro	11.42

Table 3.2: All Quantlets dealing with "YAML" available on Quantlet, lborke, b2net; extracted via **rgithubQ** and **yamldebugger**

	qlet.name	qlet.repo.path	search.score
1	blspricevec	QuantLet/SFS-ToDo	11.98
2	SFSstoploss	QuantLet/SFS	11.09
3	blsprice	QuantLet/SFE-ToDo	11.09
4	SFEItoProcess	QuantLet/SFE_class_2015	10.90
5	SFEBoundary	QuantLet/SFE_class_2015	10.90
6	SFEBoundary_V	QuantLet/SFE_class_2015	10.90
7	SFEBoundary_V_tau	QuantLet/SFE_class_2015	10.90
8	blackscholes	QuantLet/SFS	10.81
9	SFSBSCopt1	QuantLet/SFS	10.81
10	SFShullhedgeratio	QuantLet/SFS	10.76

Table 3.3: Top ten Quantlets from the Quantlet organization dealing with "black scholes", extracted via **rgithubQ** and **yamldebugger**

	Number of Quantlets
Awdesch Melzer	41
Ying Chen	14
Andreas Golle	12
Christian M. Hafner	6
Lasse Groth	6
Szymon Borak	6
Florian Schulz	5
Simon Gstöhl	5
Daniel T. Pele	4
Derrick Kanngiesser	4

Table 3.4: Top ten authors of Quantlets dealing with "black scholes", extracted via **rgithubQ** and **yamldebugger**

For the aggregation of the search results as displayed in the tables of this section, an additional parsing process of the retrieved YAML meta infos (whose locations are stored in the data structure `sr$content$items`) is necessary. This is accomplished via the lightweight parser `yaml.parser.light`, which is a function and part of the **rgithubQ** package. The

`yaml.parser.light` produces a list of parsed YAML objects, for each YAML file one "YAML list object" with YAML data fields as further list elements. Hence, the resulting list object `q_top` contains each of the 12 "YAML list objects" (in the case of the search string "yaml"). The package **yamldebugger** for extracting the YAML data fields (`yaml.getQField`) is incorporated. Finally, all information is merged into the corresponding data frames for table creation (e.g. `name_path_scores`).

3.4 Yamldebugger

3.4.1 YAML

YAML is a human friendly data serialization standard for all programming languages (`http://yaml.org/`). Designed as a human-readable and data-oriented language in 2001, YAML can easily be applied to widely used data frames such as lists and arrays. What makes YAML also rather user-friendly for maintaining hierarchical data is that it avoids the excessive use of brackets, tags and other enclosures which could make the document structure less comprehensible.

The design goals for YAML are, in decreasing priority: 1. YAML is easily readable by humans; 2. YAML data is portable between programming languages; 3. YAML matches the native data structures of agile languages; 4. YAML has a consistent model to support generic tools; 5. YAML supports one-pass processing; 6. YAML is expressive and extensible; 7. YAML is easy to implement and use.

There exist many YAML parser implementations for various programming languages, amongst them: C/C++, Java, Javascript, PHP, Python, Ruby. The R implementation is available as a package[6], which is basically a C interface to the 'libyaml', a YAML 1.1 parser and emitter, see `http://pyyaml.org/wiki/LibYAML`.

Due to the properties mentioned above, YAML was selected as annotation language for the meta information of QLs. A typical example for a YAML meta info is the text file of QuachSymanzikForsgren. Besides Quantlet, there are various GitHub organizations using YAML for metadata. Three examples with each more than 30.000 repositories are:
I) `https://github.com/GITenberg` – an open source community curating and publishing highly usable and attractive ebooks in the public domain stored as a collaborative, trackable, scriptable digital library on GitHub;
II) `https://github.com/gitpan` – a project to import the entire history of CPAN (Comprehensive Perl Archive Network) into a set of git repositories, one per distribution; and
III) `https://github.com/the-domains` – a big collection of meta information concerning web pages and their images.

3.4.2 Style Guide

The QuantNet Style Guide[7] enables a standardized audit and validation of new QLs by means of comprehensive help pages and the **yamldebugger** package, see also Section 3.4.3.

The Style Guide contains several subsections:
1) Style guide of Quantlets: an overview of the structure of a Quantlet;
2) Characteristics and mandatory data fields of the YAML meta info file *Metainfo.txt*;
3) Examples of complete and correct meta infos;
4) The main YAML rules most relevant for QuantNet;
5) Instructions on how to format the programming R code with examples of using the **formatR** package (Xie, 2016a);

[6] `https://github.com/cran/yaml`
[7] `https://github.com/Quantlet/Styleguide-and-FAQ`

6) Basic instructions for the GitHub Desktop client (`https://desktop.github.com/`);
7) Main information about the purpose of the **yamldebugger** package and further guidelines (technical terms, Quantlet repository structure, special characters etc.).

The QuantNet Style Guide was developed by several Quantlet users[8] over a longer time period and was permanently adjusted to the practical needs. Together with the **yamldebugger** and introductory yamldebugger_intro, a potential Quantlet contributor has all necessary tools for a fast, transparent and iterative code development and documentation process.

3.4.3 Yamldebugger package

In order to simplify and automate the validation process of new QLs, the YAML parser debugger package (or **yamldebugger** for short) (Borke, 2017d) was developed for testing and certifying of local versions of the GitHub repositories containing YAML metadata, see `https://github.com/Quantlet/yamldebugger` for implementation details. The yamldebugger fulfills **two main tasks**. First, it checks the Quantlet repository structure, the validity of the YAML meta information and the completeness of the mandatory data fields as described in the Style Guide, see Section 3.4.2. Second, the yamldebugger helps to analyze, standardize and unify the different YAML data fields, which are subject to varying spelling and notations.

The current yamldebugger version ranks every validated QL, thus helping to quickly identify deviations and discrepancies from the Style Guide specifications as well as YAML errors. The quality ranking system spans five different grades: "A", "B", "C", "D" and "N". "A" means the full compliance, "B" minor discrepancies, "C" more serious style violations and "D" YAML parser errors. "N" indicates that no YAML meta info could be found and must be decided on an individual basis, because a repository can contain different subfolders, those containing QLs and those without them.

```
library(yamldebugger)

workdir = "C:/GitHub/Stochastic_processes"
d_init = yaml.debugger.init(workdir, show_keywords = FALSE)
qnames = yaml.debugger.get.qnames(d_init$RootPath)
d_results  = yaml.debugger.run(qnames, d_init)
( OverView = yaml.debugger.summary(qnames, d_results, summaryType = "compact") )
```

Listing 3.2: The interaction of the four main functions of the **yamldebugger** package

Listing 3.2 demonstrates the interaction of the four main functions of the **yamldebugger**. This set of functions is responsible for the first main task (validity of the YAML meta information and repository structure). Listing A.1 demonstrates the practical application, using the repository Stochastic_processes as an example.

```
subset(OverView, !(`Q-Quali` %in% c("A")) )
yaml.not.Qdfields(d_results$meta_names_distribution)
rowSums(sapply( d_results$Metainfos, function(yaml)
              { yaml.Qdfields.nchar.from.meta(yaml) } ))
d_names = unlist(sapply( d_results$Metainfos, function(yaml)
              { yaml.Qdfields.from.meta(yaml)$found_dnames } ))
( d_names_distr = sort(table(d_names), decreasing = TRUE) )
```

Listing 3.3: YAML data field analysis via **yamldebugger** functions

Listing 3.3, on the other hand, illustrates the interplay of the **yamldebugger** functions for YAML data field analysis. The corresponding example is given in Listing A.2 using the same

[8] `https://github.com/Quantlet/Styleguide-and-FAQ/graphs/contributors`

QLs for validation as in Listing A.1. Diverse characteristics as quality ranking grades, distributions and occurrences of data field names, their validity, their character distributions etc. can be aggregated, analyzed, evaluated, and, if necessary, the YAML data field matching list `Q_dfield_list`[9] from the package itself can be adjusted. A meaningful and reasonable calibration of this matching list is crucial for further extraction of YAML data fields (via the `yaml.getQField` function of the **yamldebugger** package) within the TM and cluster validation steps, as will become apparent in Sections 3.6.3 and 3.7.1. The function `yaml.getQField` was already used in Listing 3.1.

The function `yaml.debugger.summary` in Listing 3.2 allows three different levels of summary details: `mini`, `compact` and `full`. Together with the R package **knitr** (Xie, 2016b), the yamldebugger summary can be easily converted into a GitHub compliant Markdown table via `kable(OverView)`. Since the **yamldebugger** version 1.0 all validated Quantlet repositories contain the file "yamldebugger_results.md", see e.g. the SFE repository[10]. This reporting process can be even simplified by means of the package **git2r** (Widgren and others, 2016).

The introductory QLs yamldebugger_intro provide more examples on how to install and run the **yamldebugger** with additional analysis and visualization capabilities, see also Figures A.2 and A.3. The yaml_TDM_CorrPlot visualizes correlations between the most frequent keywords of the document-term matrix, which is extracted from the keywords in the Quantlet YAML meta infos, see Figure A.2. Listing 3.4 shows the relevant code part from the yaml_TDM_CorrPlot.

```
> DTM
<<DocumentTermMatrix (documents: 1198, terms: 980)>>
Non-/sparse entries: 11383/1162657
Sparsity           : 99%
Maximal term length: 35
Weighting          : term frequency (tf)
> DTM_graph = plot(DTM, terms = findFreqTerms(DTM, lowfreq = 60), corThreshold = 0.2, weighting
    = TRUE)
> DTM_graph
[1] "A graph with 37 nodes."
```

Listing 3.4: yaml_TDM_CorrPlot application example

[9] https://github.com/lborke/yamldebugger/blob/master/R/yaml.Qdfields.R
[10] https://github.com/Quantlet/SFE/blob/master/yamldebugger_results.md

3.5 Google Analytics

The aim of this section is to provide insights into the automatized integration of data from *Google Analytics* into R for further processing and analysis. The data of interest are various download statistics of the QuantNet website. Within the *Collaborative Research Center 649: Economic Risk*, there was a regular meeting in which a certain set of statistics was presented. These presentations were created on a monthly basis and in a specific build-up. Their content however needed to be updated each time based on the log files of the Linux server. Originally, this was done manually in *Excel* and then exported to *PowerPoint*. Later, this was adopted into the R environment by means of the *Google Analytics API*, such that up-to-date statistics are available through a program in R.

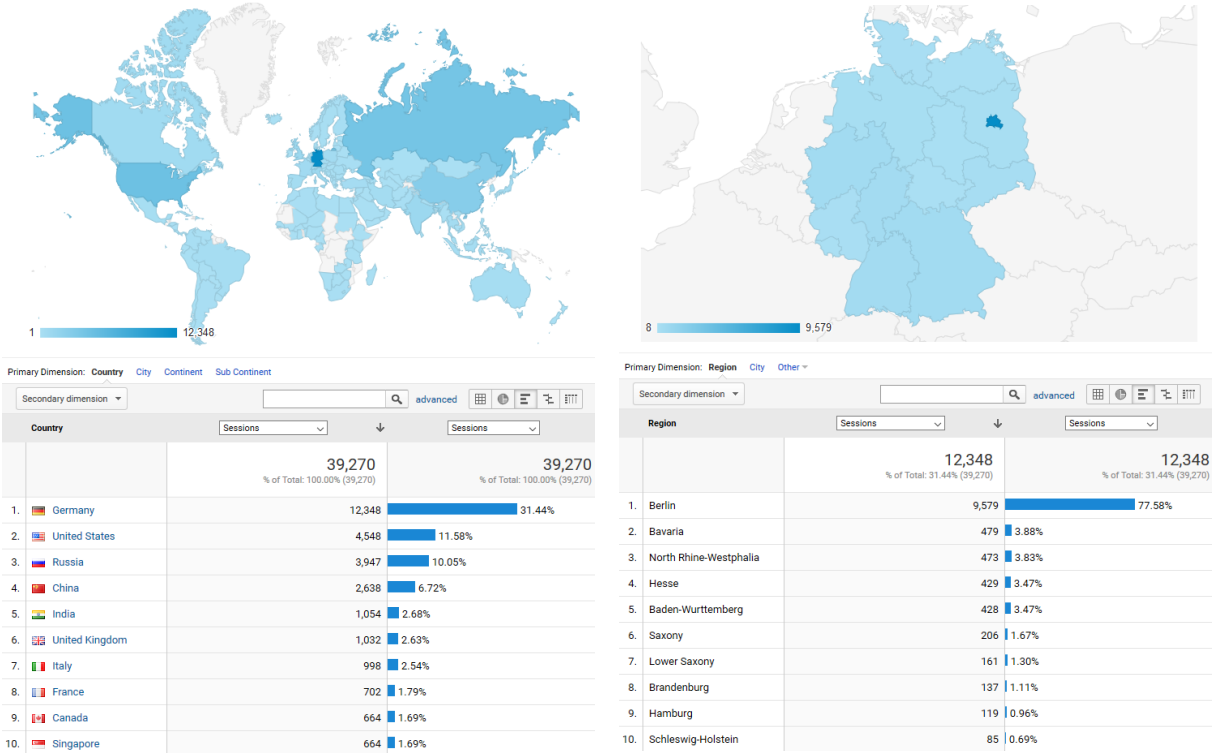


Figure 3.2: QuantNet visitors via *Google Analytics*: global view (left), Germany(right)

3.5.1 Introduction to Web Metrics

Web metrics, also known as *web analytics*, is the process of collecting, analysing and reporting online traffic generated by internet users on a website. This can be helpful for improving the usability of a particular website or to get valuable information about the relevance of the provided content. There are several tools to analyse the web traffic which differ in their functionalities and complexity. In a nutshell, these tools cover combinations of browser logging tools and user panels, the collection of network traffic data provided via the Internet service provider as well as site-specific server log parsers or page tagging technologies.

Google Analytics[11] represents the latter. It is a page-tagging tool, which employs first party cookies to track user behavior. This means, that user data is collected via the web browser and sent to a remote data-collection server. Google provides this service and the relevant reports

[11] `https://www.google.com/analytics/`

for further analysis. Kaushik (2010) delivers a good introduction into the field of online data mining and *predictive analytics*. Prem et al. (2016) present various indicators (among them *Google Analytics*) to measure open science implementations and to create an Open Science Observatory (`http://opendigitalscience.eu`).

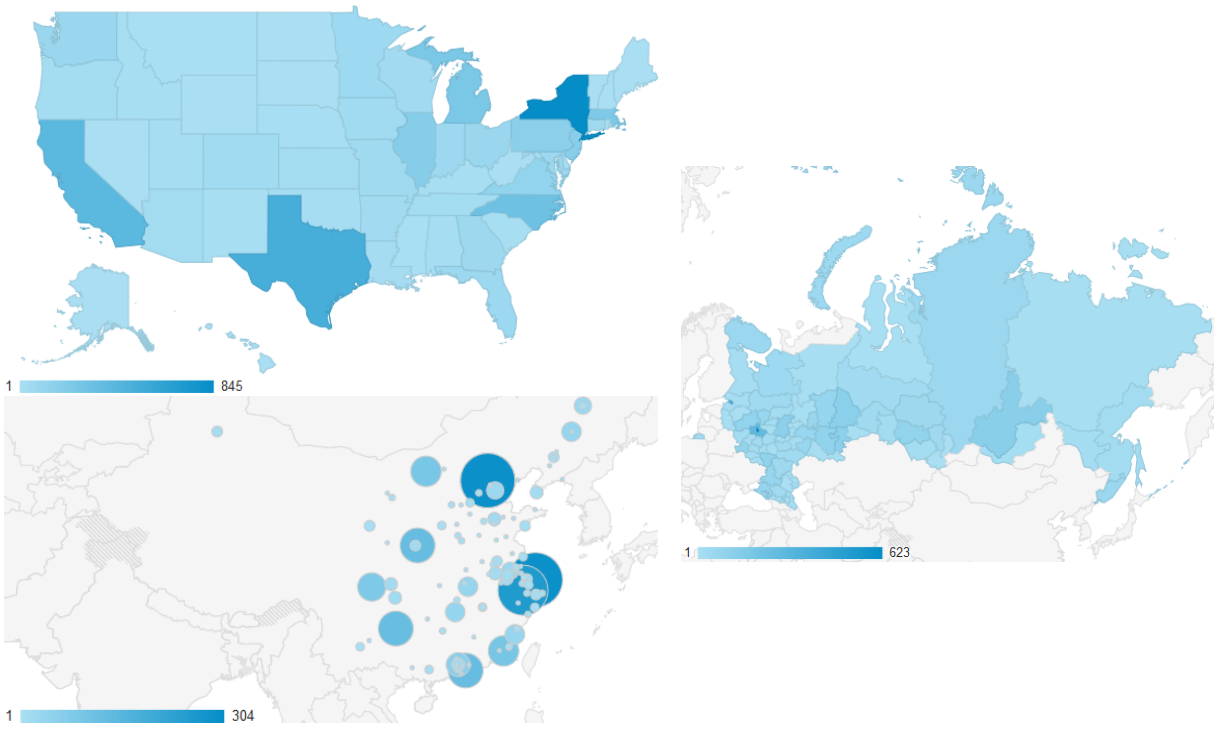


Figure 3.3: QuantNet visitors from USA (upper left), Russia (right) and China (lower left)

Figures 3.2 and 3.3 show the *Audience Overview*, which is accessible via the *Google Analytics* website. This reporting tool allows manual configuration and parametrization of the desired web analytics results. In the given case, the total user sessions in the time period 16.11.2013 – 18.11.2016 are grouped and sorted by countries. The most visitors come from Germany (Figure 3.2), followed by those from the USA, Russia and China (Figure 3.3).

3.5.2 RGoogleAnalytics in a nutshell

Thanks to the **RGoogleAnalytics** package (Pearmain et al., 2014), R possesses a *Google Analytics API* binding. Therefore, it is possible to access and query data from *Google Analytics* directly. Additionally, the package enables access to all *Google Analytics* accounts of a user. In terms of this section, all download and user behavior statistics for the QuantNet website can be acquired conveniently through that package[12].

The **RGoogleAnalytics** package allows for integration of data from *Google Analytics* into the R environment. Six functions are included, which will be presented here. Because *Google Analytics* and R are two different application environments, a connection needs to be established for further processing of any kind of data.

```
oauth_token <- Auth(client.id = "XXX", client.secret = "YYY")
save(oauth_token, file = "oauth_token")
load("oauth_token")
ValidateToken(oauth_token)
```

[12] QuantNet first created a *Google Analytics* account on 16th of November 2013 and all statistics can only be retrieved starting then

The `Auth` function serves that purpose and is the necessary first step. Precisely speaking, it authorizes the **RGoogleAnalytics** package to the user's *Google Analytics* account using *OAuth2.0*, an open protocol for standardized and secure *API*-authorization between applications. Two arguments are required: `client.id` resembles the user name and `client.secret` the corresponding pass-phrase. The created token can be saved to a file and, in subsequent runs, called up again without requiring the user's consent. Only if the user queries another *Google Analytics* profile with another email account, that consent is required again. However, that token has a 60 minute lifetime, after which a new token can be obtained by using the `ValidateToken` method. This method checks if the token is expired, and if this is true, a new token will be generated and the token object updated.

```
GetProfiles(oauth_token)
```

Continuing from there, the `GetProfiles` function creates a data frame of all profile IDs and profile names by using the created token as the only argument. After retrieving all profile information, *Google Analytics* query parameters need to be initialized.

```
query.params.list <- Init(start.date = NULL, end.date = NULL, dimensions = NULL,
    metrics = NULL, filters = NULL, sort = NULL, segments = NULL, max.results = NULL,
    start.index = NULL, table.id = NULL)
```

This can be done by the `Init` function. It combines all query parameters into a list. Parameters like `start.date` and `end.date` define the timeframe for the requested *Google Analytics* data. Up to 7 dimensions can be set by the `dimensions` argument and up to 10 metrics by the `metrics` argument. In both cases this can be done as a single `string` or as a vector of `strings`. Besides setting the scope, several arguments can be used to preprocess the retrieved data. With `sort`, the sorting order of the returned data can be set. Additionally, a filter string for the *Google Analytics* request can be used to narrow the data for processing. The `segments` argument serves the purpose to slice and dice the data to define segments. Finally, the `start.index` and `max.results` arguments set the first row and the following number of rows, which will be included in the query response. The `table.id` depicts the *Analytics View ID*, for which the query will retrieve the data. All query parameters are `NULL` by default. Listing 3.5 demonstrates the use of the parameters. After initializing the query list, we pass it on to the `QueryBuilder` function.

```
ga.query <- QueryBuilder(query.params.list)
```

The function `QueryBuilder` initializes an object with all query parameters and validates them. The created object `qa.query` in combination with the created token are passed on to the `GetReportData` function.

```
GetReportData(ga.query, oauth_token, split_daywise = FALSE, paginate_query = FALSE)
```

Additional optional arguments like `split_daywise` and `paginate_query` are available: the first one splits the query into day-wise partitions by date range, the second one numbers chunks of results by requesting a maximum number of allowed rows at a time. Finally, the `GetReportData` function retrieves the requested data from the *Core Reporting API*.

3.5.3 Metrics, Dimensions, Event Tracking in Google Analytics

In general, a metric is a quantitative measurement of statistics describing events or trends on a website. A key performance indicator (KPI) is a metric that helps to understand how you are doing against your objectives (Kaushik, 2010).

In *Google Analytics*, *metric* is a number, which is used to measure one of the characteristics of a dimension. A *dimension* is the attribute of visitors to a given website. Taken together,

a dimension provides context to a metric. Though both dimensions and metrics are the characteristics of the website visitors, they are different in the way they are configured, collected, processed, reported and queried in *Google Analytics*.

Event Tracking captures data differently from the standard tag-based Page View data. The event data is stored differently and creates new metrics that capture the unique experience of rich media and user actions triggered by click events or by keyboard entries. The *Event Tracking - Dimensions and Metrics Reference* describes all event tracking dimensions and metrics available in the *Real Time Reporting API*.

3.5.4 Most downloaded Quantlets: a code example

The presented script in Listing 3.5 retrieves the most downloaded QLs. It extracts recent download statistics from *Google Analytics* starting November 2013 for each QL. Furthermore, short explanations (based on the desciptions in the meta information) of the specific QLs are added and the final results are presented as an R data frame. The API query specifies the parameters `dimensions`, `metrics` and `filters` conditioning the desired *Event Tracking* criteria. Every time the user clicks on a Quantlet page, this interaction is tracked by *Event Tracking* and is available for further analysis. The R code for the full reproducibility with additional postprocessing (short explanations) and LaTeX-table output is available as TopDownloads.

```
query.list.Qlet <- Init( start.date = "2013-11-16", end.date = "today",
       dimensions = "ga:eventLabel", metrics = "ga:totalEvents",
       filters = "ga:eventCategory==QNetShow", sort = "-ga:totalEvents",
       max.results = 2000, table.id = "ga:78690351")
ga.query <- QueryBuilder(query.list.Qlet)
ga.df <- GetReportData(ga.query, oauth_token)

samplesize = 8
colnames(ga.df) = c("Quantlet", "Downloads")
# output as data frame, top downloaded Quantlets as defined by samplesize
ga.df[1:samplesize,]
```

Listing 3.5: RGoogleAnalytics code for extracting the most downloaded Quantlets

Quantlet	Downloads
autocorr.m (autocorrelation plots)	2450
MVACARTBan1 (US bankruptcy analysis)	677
SFSmeanExcessFun (generalized Pareto distribution)	393
SFEVolSurfPlot (implied volatility surface)	371
MVAandcur (Andrew's curves)	363
SMSboxcar (Boxplot car mileage)	339
blsprice (Black-Scholes price function)	331
IBTblackscholes (call & put options - Black Scholes)	327

3.5.5 Most Quantlet downloads by country: a code example

Listing 3.6 shows only the required changes relative to the code in Listing 3.5, see DownloadsByCountry for the full code. Within the `Init` function, only the parameter `dimensions` needs to be adjusted. While the metric and retrieved raw data remain the same, the final information is aggregated by the new dimension "country". The remaining two changes concern the format of the output data frame. It should be noted that the web analytics results from Listing 3.6 differ from those in Figure 3.2. The first results reflect the download statistics

triggered by user events (mouse click), the latter reflect the total user sessions (as defined in *Audience Overview*) in the same time period. This is exactly the difference between the *Event Tracking* and *Page View* approach, see Section 3.5.3. During the same session a user can perform several user events like mouse clicks.

```
dimensions = "ga:country" # in the 'Init' function
samplesize = 10
colnames(ga.df) = c("Country", "Downloads")
```

Listing 3.6: RGoogleAnalytics code for extracting the most Quantlet downloads by country

Country	Downloads
Germany	37042
United States	9735
China	9266
Bulgaria	2502
Russia	2356
United Kingdom	2265
India	1653
Italy	1643
Japan	1609
France	1335

3.5.6 Most frequent search queries: a code example

The code example in Listing 3.7 demonstrates the main part of the procedure how to retrieve the most frequent search queries entered into the search field of the QuantNet visualization. See the complete QTopSearch for all details. The R code uses the same `dimensions` and `metrics` parameters as in Listing 3.5. The main difference are other `filters` (`eventCategory = QNet2Visu` and `eventAction = search`) and `table.id` settings. The latter parameter specifies from which *Google Analytics view* (profile) to retrieve data. The QuantNetXploRer in Figure 3.4 is the "source" for events which are observed by the profile with the `table.id` = 134092861.

```
query.list.search <- Init( start.date = s_date, end.date = "today",
       dimensions = "ga:eventLabel", metrics = "ga:totalEvents",
       filters = "ga:eventCategory==QNet2Visu;ga:eventAction==search",
       sort = "-ga:totalEvents", max.results = 1000, table.id = "ga:134092861")
ga.query <- QueryBuilder(query.list.search)
ga.df <- GetReportData(ga.query, oauth_token)

samplesize = 20
colnames(ga.df) = c("SearchQuery", "frequency")
ga.df[1:samplesize,]
```

Listing 3.7: RGoogleAnalytics code for extracting the most frequent search queries

As we will see in the next section, the most frequent search queries obtained from *Google Event Tracking metrics* can be used as queries in the *test collections* in order to evaluate and calibrate the information retrieval (IR) performance in different TM models. In this manner, we have a kind of "Google Analytics driven" dynamic test queries for IR evaluation allowing to concentrate on the most frequent and hence most popular search queries, which could help to reduce the efforts usually encountered in IR relevance assessments.

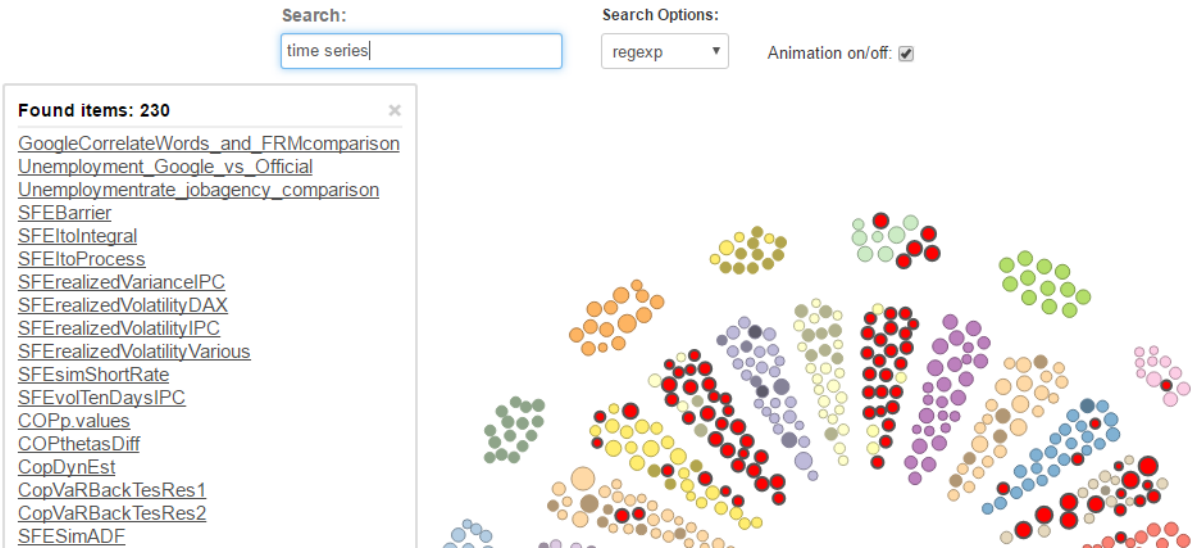


Figure 3.4: Search field in the QuantNetXploRer: after every keystroke the Quantlets relevant to the search query are displayed both in textual form as in graphical form; additionally the search queries are tracked via Google Analytics

3.6 IR in a nutshell

An examination of the opening pages of a number of information retrieval (IR) books reveals that each author defines the topic of IR in different ways. Some say that IR is simply a field concerned with organizing information (Salton, 1968), and others emphasize the range of different materials that need to be searched (Witten et al., 1999). While others stress the contrast between the strong structure and typing of a database system with the lack of structure in the objects typically searched in IR (Rijsbergen, 1979). Across all of these definitions, there is one thing in common: IR systems have to deal with incomplete or underspecified information in the form of the queries issued by users. The IR systems must be able to handle the users' underspecified query.

The typical interaction between a user and an IR system has the user submitting a query to the system, which returns a ranked list of objects that hopefully have some degree of relevance to the user's request with the most relevant at the top of the list. Sections 3.3 and 3.6.5 provide some examples. The success of such an interaction is affected by many factors, the range of which has long been considered. Sanderson (2010) discusses the following five:

1. The ability of the system to present all relevant documents
2. The ability of the system to withhold non-relevant documents
3. The time interval between the demand being made and the answer being given
4. The physical form of the output (i.e., presentation)
5. The effort, intellectual or physical, demanded of the user

Sanderson (2010) points out that many other factors can be considered, some of them are: 1) the ability of the user at specifying their need, 2) the interplay of the components of which the search algorithm is composed, 3) the type of user information need.

3.6.1 Test Collections

A strong focus of IR research has been on measuring the effectiveness of an IR system: determining the relevance of items, retrieved by a search engine, relative to a user's information

need. The vast majority of published IR research assessed effectiveness using a resource known as a *test collection*, applying thereby evaluation measures. There are many conferences and meetings devoted purely to *test collections*, including three international conferences, TREC, CLEF, and NTCIR, which together took place more than 30 times since the early 1990s. This research focus is a part of a longer tradition which was motivated by the creation and sharing of testing environments in the previous three decades, which itself was inspired by innovative work conducted in the 1950s.

The classic components of a *test collection* are as follows: 1) a *collection of documents*, 2) a *set of topics* (also referred to as queries), 3) a *set of relevance judgments* composed of a list of topic/document pairs, detailing the relevance of documents to topics.

The genesis of IR evaluation is generally seen as starting with the work of Cleverdon and his Cranfield collections, built in the early 1960s. However, he and others were working on retrieval evaluation for most of the 1950s (Cleverdon, 1959). In his first collection Cleverdon tested four competing indexing approaches on a set of 18.000 papers. The papers were manually indexed using each of the four classification methodologies. Once the indexes were built, the papers were searched with 1.200 "search questions". The collection became known as Cranfield I. Cleverdon concluded that the relatively large size of Cranfield I was not important in ensuring reliable measurements. Therefore, the new collection was composed of 1.400 "documents" (titles, author names and abstracts) derived from the references listed in around 200 recent research papers. This work resulted in the Cranfield II collection comprising 1.400 documents, 221 topics, and a set of complete variable level relevance judgments. Alongside the work of Cleverdon, Salton initiated the creation of a series of test collections, collectively known as the SMART collections (Lesk and Salton, 1968). A good review of the IR research and its history is provided in Sanderson (2010).

3.6.2 Effectiveness measures: Precision, Recall

	Relevant	Non-relevant	
Retrieved	a	b	$a+b$
Not retrieved	c	d	$c+d$
	$a+c$	$b+d$	$a+b+c+d$

Table 3.5: Contingency table of all possible quantities in IR

Cleverdon et al. (1966) produced a *contingency table* of all possible quantities that could be calculated to judge an information retrieval system. Table 3.5 is reproduced including the original labels. Another common notation for the table's cells in Table 3.5 is: *true positives* (a), *false positives* (b), *false negatives* (c) and *true negatives* (d), see Manning et al. (2008). Various measures can be created out of combinations of the table's cells. The three that are probably the best known are:

$$\text{Precision} = \frac{a}{a+b}\ , \qquad \text{Recall} = \frac{a}{a+c}\ , \qquad \text{Fallout} = \frac{b}{b+d}\ . \tag{3.1}$$

Where *precision* measures the fraction of retrieved documents that are relevant, *recall* measures the fraction of relevant documents retrieved and *fallout* measures the fraction of non-relevant documents retrieved. Most IR systems experience a dilemma concerning them. To improve a system's precision, the system needs strong measures for deciding whether a document is relevant to a query. This will help minimize the *false hits*/false positives (quantity b in Table 3.5), but it will also affect the number of relevant documents that are retrieved. These strong measures can

prevent some important relevant documents from being included within the set of documents that satisfy the query, thereby lowering the recall (Herbert et al., 2004). Further, because the number of relevant documents in a set of documents is fixed for any query (it is the quantity $a + c$ in Table 3.5) we can use the true positives (quantity a) as proxies for the "relative recall". In practice the determination of the total number of relevant documents relative to a query is difficult and time-consuming due to the size of the document corpus. A higher true positive value implies a higher recall value, up to the scalar multiple of $\frac{1}{a+c}$.

3.6.3 Google Analytics driven QuantNet Test Collection

As data set for the following IR performance analysis, the current YAML meta information was parsed from QuantNet by means of the TM pipeline, see Figure 3.1. Hence, our *collection of documents* contains 1140 YAML documents. Our *set of topics* was formed from the most frequent search queries delivered by the Google Event Tracking metrics, see Section 3.5.6. The *set of relevance judgments* is determined on demand, when additional precision and recall benchmarks are required. All results and tables were calculated by use of the **TManalyzerQ** package, which is the TM layer of the "GitHub API based QuantNet Mining infrastructure in R".

Throughout the rest of our article we will use the definitions and notations from Section **Vector space representations** in Borke and Härdle (2016). The most important quantities are: $Q = \{d_1, \ldots, d_n\}$ as a set of documents/Quantlets; $T = \{t_1, \ldots, t_m\}$ as a dictionary (set of all terms); $tf(d, t)$ as the absolute frequency of term $t \in T$ in $d \in Q$; $tf\text{-}idf$ as the term frequency - inverse document frequency; D as a "term by document matrix" TDM. The TM models BVSM, GVSM(TT) and LSA were considered. From LSA two configurations were examined: LSA (50% of the weight of all singular values maintained) and LSA50 (with the dimension parameter $k = 50$).

3.6.4 IR system designs in 3 models

All IR results and TDM representations in this section were calculated by means of the **TManalyzerQ** package as displayed in Listing A.3.

Single term queries

	q1	q2	q3	q4	q5
covar	1	0	0	0	0
histogram	0	0	0	1	0
multivari	0	0	0	0	1
quantil	0	0	1	0	0
random	0	1	0	0	0

Table 3.6: TDM of the single term queries in the post processed raw TF-form

Via **RGoogleAnalytics** (see Listing 3.7) the following most frequent single term search queries were obtained: "covar", "random", "quantile", "histogram", "multivariate", see Table 3.6. They serve as the first *set of topics* in the IR simulation in Listing A.3. A document is retrieved if its similarity value relative to the query is bigger than the given IR threshold. Taking different weighting schemes (argument `tf_weight`) and varying threshold levels (argument `sim_threshold`), we obtain the IR results as summarized in Table 3.7.

The results in Table 3.7 can be summarized as follows. The LSA50 model retrieves the most documents for all `tf_weight` × `sim_threshold` combinations. BVSM returns the least number of matches. Under the weighting scheme tf no clear dominance between GVSM(TT) and LSA can be determined. Under the weighting scheme *tf-idf* GVSM(TT) outperforms LSA in many cases. Additionally, the weighting scheme *tf-idf* retrieves more matches averaged over all TM models × `sim_threshold` combinations.

	B	TT	LSA	L50	B	TT	LSA	L50	B	TT	LSA	L50
weighting scheme	tf normalized											
covar	0	0	0	7	0	0	3	9	0	0	6	9
random	0	0	0	6	0	1	0	7	0	10	3	18
quantile	0	0	0	0	0	1	0	1	0	1	2	6
histogram	0	0	0	3	0	0	2	6	2	2	4	14
multivariate	0	0	0	0	0	0	0	4	0	0	0	16
weighting scheme	tf-idf normalized											
covar	0	0	2	7	0	4	5	9	0	6	6	11
random	0	0	0	11	0	3	2	17	0	13	9	24
quantile	0	0	0	0	0	0	0	1	0	3	2	17
histogram	0	2	1	13	1	10	7	19	3	16	13	26
multivariate	0	0	0	5	0	2	0	12	0	12	0	21

Table 3.7: Number of QLs retrieved in each of 3 TM models (single term queries); measure: cosine similarity; similarity threshold for IR: 0.8, 0.7, 0.6 (from left to right)

Compound term queries

	q1	q2	q3
black	0	0	1
multivari	0	1	0
number	1	0	0
random	1	0	0
schole	0	0	1
statist	0	1	0

Table 3.8: TDM of the compound term queries in the post processed raw TF-form

Via **RGoogleAnalytics** (see Listing 3.7) the following most frequent compound term search queries were obtained: "random number", "multivariate statistics", "black scholes", see Table 3.8. The second *set of topics* for the IR simulation in Listing A.3 yields the results presented in Table 3.9.

The results in Table 3.9 show a similar situation as in the case of "single term queries". But GVSM(TT) is clearly better than LSA. LSA50 is best and BVSM is still worst (concerning the number of hits). One remarkable observation in all Tables (3.7 and 3.9) is that LSA50 clearly outnumbers the other models at the highest IR threshold = 0.8. The other models have none or very few hits.

	B	TT	LSA	L50	B	TT	LSA	L50	B	TT	LSA	L50
weighting scheme	tf normalized											
random n.	0	1	0	7	0	8	2	7	1	8	6	15
multivariate s.	0	0	0	0	0	1	0	6	0	11	0	11
black s.	0	1	0	20	0	44	0	46	0	58	1	55
weighting scheme	tf-idf normalized											
random n.	0	1	0	9	0	6	1	17	1	14	5	23
multivariate s.	0	0	0	6	0	4	0	15	0	14	0	22
black s.	0	3	0	43	0	43	1	51	0	52	1	59

Table 3.9: Number of Qs retrieved in each of 3 TM models (compound term queries); measure: cosine similarity; similarity threshold for IR: 0.8, 0.7, 0.6 (from left to right)

3.6.5 IR Performance: recall and precision in 3 models

In order to assess the IR effectiveness and validity of the previous results, we have to examine the relevance of the retrieved documents for each query individually. This is demonstrated by two examples.

```
# Single term queries
query = c("covar", "random", "quantile", "histogram", "multivariate")

query.tm.folded = query.tm.fold_in(query, tm_list, tf_weight = "ntc")
q_tdm_sim.tm_res = q_tdm_sim.tm.list(query.tm.folded)
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.8)
q_ir_list$query_tm_text["histogram"]
# returns the retrieved docs for every TM model, see below for the full output
"BVSM: no hits\\GVSM(TT): SPMHistoConstruct (0.9), SPMhistobias2 (0.82)\\LSA: SPMHistoConstruct
    (0.9)\\LSA50: SPMhistobias2 (0.97), SPMHistoConstruct (0.96)..."

q_ir_list$query_tm_list[["histogram"]]
 [1] "SPMHistoConstruct" "SPMhistobias2" "BCS_hist1" "BCS_HistBinSizes" "BCS_hist2"
 [6] "SPMbuffagrid" "SPMbuffahisto" "SPMhistogram" "SPMashstock" "SPMstockreturnhisto"
 [11] "SPMhiststock" "SPMsimulatedexponential" "SPMbuffadata"
```

Listing 3.8: IR results inspection via **TManalyzerQ**

The extended IR results for the search query "histogram" (tf-idf, IR threshold 0.8, numbers in brackets show the similarity values) are produced by the function `query.similar.doc.inspect` and stored in the variable `q_ir_list$query_tm_text`, see Listing 3.8:

BVSM: no hits;
GVSM(TT): SPMHistoConstruct (0.9), SPMhistobias2 (0.82);
LSA: SPMHistoConstruct (0.9);
LSA50: SPMhistobias2 (0.97), SPMHistoConstruct (0.96), BCS_hist1 (0.95), BCS_HistBinSizes (0.92), BCS_hist2 (0.91), ~~SPMbuffagrid~~ (0.9), SPMbuffahisto (0.89), SPMhistogram (0.89), SPMashstock (0.89), SPMstockreturnhisto (0.87), SPMhiststock (0.87), SPMsimulatedexponential (0.85), ~~SPMbuffadata~~ (0.82)

Manual inspection of the retrieved QLs for the query "histogram" shows that there are two *false hits* (SPMbuffagrid, SPMbuffadata) in the LSA50 model. The other TM models have maximum precision (i.e. $Precision = 1$), whereas the precision of LSA50 is $\frac{11}{13}$. The (relative) recall performance is: LSA50 $>_{recall}$ GVSM(TT) $>_{recall}$ LSA $>_{recall}$ BVSM, see Section 3.6.2 for IR effectiveness quantities and notations.

The IR results for the search query "random number" (tf-idf, IR threshold 0.7) were produced in an analogous way as in Listing 3.8:

BVSM: no hits;
GVSM(TT): SFEfibonacci (0.86), SFErandu (0.79), SFErangen2 (0.76), SFErangen1 (0.75), random_walk (0.74), SFEBMuller (0.71);
LSA: SFEfibonacci (0.78);
LSA50: BCS_LFG (0.97), SFErandu (0.95), SFEfibonacci (0.95), SFErangen2 (0.95), SFErangen1 (0.94), BCS_RANDU (0.9), BCS_ARM (0.86), BCS_Shapes (0.85), random_walk (0.81), SFEBMuller (0.8), SFEevt3 (0.78), randomwalk_ar1 (0.77), simulationplot (0.76), SFEtrinomp (0.72), MSMasprob (0.72), SFEevt2 (0.72), BCS_claytonMC (0.7)

Manual inspection of the retrieved QLs for the query "random number" shows that all hits are relevant. Hence, we can conclude that all TM models provide maximum precision, i.e. all retrieved documents are relevant, and: LSA50 $>_{recall}$ GVSM(TT) $>_{recall}$ LSA $>_{recall}$ BVSM.

```
false_hits = list("histogram" = c("SPMbuffagrid", "SPMbuffadata"))

q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.8,
                                      false_hits = false_hits)

m_retr = q_ir_list$retrieved_m
m_true_positives = q_ir_list$relevant_m
m_precision = q_ir_list$relevant_m / q_ir_list$retrieved_m
m_precision[is.nan(m_precision)] = 0

colnames(m_retr) = colnames(m_true_positives) = colnames(m_precision) =
                   c("B", "TT", "LSA", "L50")
( m_IR = cbind(m_retr, m_true_positives, round(m_precision, 2)) )
```

Listing 3.9: IR effectiveness inspection via **TManalyzerQ**

The manual inspection of all single term queries' results has revealed that there are two *false hits* (tf-idf, IR threshold 0.8). Incorporating this information into the function `query.similar.doc.inspect` (see Listing 3.9) allows to build three matrices as shown in Table 3.10: $M_{retrieved}$, $M_{true\_positives}$, $M_{precision}$. We have thus for each query × TM model combination the number of retrieved documents, the number of retrieved and relevant documents (true positives) and the precision value. We can conclude that the number of true positives for all single term queries is maximal in the LSA50 model. Except the combination "histogram"/LSA50 all other cases (query × TM model combinations) show maximum precision.

	B	TT	LSA	L50	B	TT	LSA	L50	B	TT	LSA	L50
	$M_{retrieved}$				$M_{true\_positives}$				$M_{precision}$			
covar	0	0	2	7	0	0	2	7	0	0	1	1.00
random	0	0	0	11	0	0	0	11	0	0	0	1.00
quantile	0	0	0	0	0	0	0	0	0	0	0	0.00
histogram	0	2	1	13	0	2	1	11	0	1	1	0.85
multivariate	0	0	0	5	0	0	0	5	0	0	0	1.00

Table 3.10: IR performance for single term queries, tf-idf, IR threshold 0.8: $M_{retrieved}$, $M_{true\_positives}$, $M_{precision}$ (from left to right)

3.7 Cluster Validation

Clustering plays a major role in dealing with high-dimensional data. Being first used for simple classifications, clustering ended up as an integral part of different disciplines like archeology, linguistics, bioinformatics, genetics and others (Everitt et al., 2011). Nowadays a vast amount of various numerical methods are introduced in order to make the retrieval of information from the clustering partition easier and more efficient and also to assess how efficient it is.

Our main goal here is to try different clustering and validation methods within the R software environment and to determine the optimal combination of data configuration, vector space model, clustering method and clustering criteria settings, thereby evaluating the resulting partition and eventually finding a reasonable number of clusters. The so-called "5-level Validation Pipeline" is described in Section 3.7.1.

Depending on the matrix P (an appropriate linear transformation), we will consider three TM models, examined in Cristianini et al. (2002) and evaluated in the M^3-benchmark as described in Borke and Härdle (2016): BVSM, GVSM(TT) and LSA. Furthermore, we will compare three different clustering methods: hierarchical clustering, k-means and k-medoids (or pam - partitioning around medoids). All of them are rather well-known and often used, see for more details Everitt et al. (2011).

For validation of clustering methods different measures (also called clustering criteria, clustering validity indices or quality indices) were introduced. We will consider those of them implemented in the R packages **clusterCrit** (Desgraupes, 2016) and **NbClust** (Charrad et al., 2014). The optimal number of clusters can be derived by maximizing/minimizing of the index value or the difference between two successive slopes. The last means that on a plot with index values Q against the number of selected clusters $K \in \{K_m, \ldots, K_M\}$, the best value for K corresponds to an elbow. Suppose, for example, we need to maximize the difference between two successive slopes. Let us denote $V_i = Q_{i+1} - Q_i$, then K is determined by:

$$K = \arg \max_{K_m < i < K_M} (V_i - V_{i-1}).$$

Some of the 27 internal quality indices offered by the package **clusterCrit** are of the exceptional interest, allowing rather clear interpretation. The following list contains the names (as used in Desgraupes (2013)) of 12 measures that we have selected for our benchmark: Ball-Hall, C-Index, Calinski-Harabasz, Davies-Bouldin, Dunn, McClain-Rao, Ratkowsky-Lance, Ray-Turi, Silhouette, Trace-W, Wemmert-Gancarski and Xie-Beni.

All these clustering validity indices combine information about intracluster compactness and intercluster isolation, as well as other factors, such as geometric or statistical properties of the data, the number of data objects and dissimilarity or similarity measurements. More about the theory, index formulas and additional information can be found in Brock et al. (2008), Desgraupes (2013) and Charrad et al. (2014).

3.7.1 Validation Pipeline

The experimental design for cluster validation presented here is a direct evolution of the $M^3_{d_1,d_2,d_3,max}$ design as introduced in Borke and Härdle (2016). Additionally, we embed another new dimension into the concept, namely different configurations of meta information, thus expanding $M^3_{d_1,d_2,d_3,max}$ to $M^4_{d_1,d_2,d_3,d_4,max}$. The new M stands for the new dimension "meta information". Let us call this performance validation approach the **Validation Pipeline** (*Vali-PP*). The main goal of this validation benchmark is to calibrate such a combination of all 5 dimensions (*Vali-PP*-configurations), that provides the best clustering result, when applied to YAML data. An intuitive interpretation of this idea could be a pipe with 5 gear wheels (each

of them representing one optimization component/parameter), that takes preprocessed (smart) data as an input and returns an optimal smart clusterization as an output. A schematic illustration of the described process is displayed in Figure 3.5.

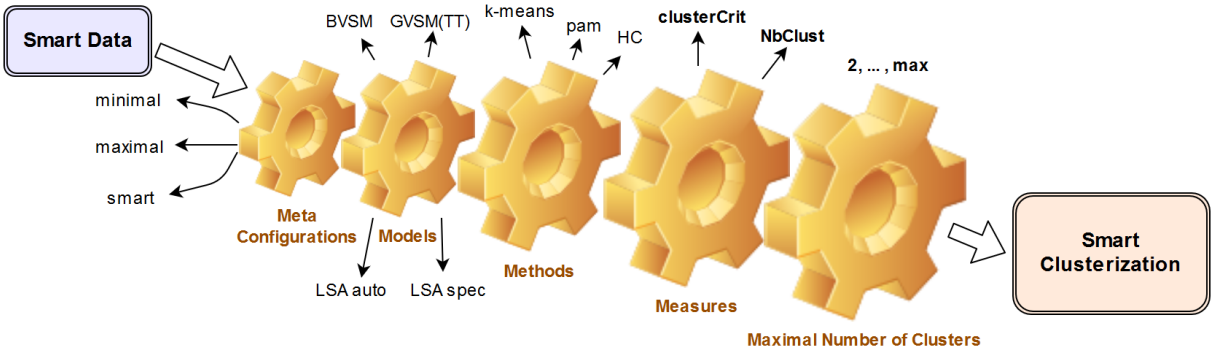


Figure 3.5: Validation Pipeline $M^4_{d_1,d_2,d_3,d_4,max}$

The main purpose of this analogy is to determine the best angle of rotation for each gear wheel, so that together their combination lets smart data pass through the pipe in the most effective way. The term "effectiveness" means here that we should try to find a compromise between the information gain from our data and the dimension reduction of data at the same time, omitting unnecessary information and hence reducing the storage and computational costs.

Vali-PP in a nutshell

Summarized, the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$ deals with the following 5 dimensions:

- d_1: 3 sample configurations of **meta information**: minimal, maximal, smart
- d_2: 3 vector space **models** / 4 TM configurations: BVSM, GVSM(TT), LSA (automatic choice of the dimension and own choice of the dimension, e.g. LSA25), encoded in "standard TM colors": BVSM, GVSM(TT), LSA, LSA25
- d_3: 3 clustering **methods**: k-means, k-medoids, Hierarchical Clustering (HC) (comprising average, ward.D and ward.D2 agglomeration methods)
- d_4: 12 **measures** (quality indices) for validating the clustering quality (from R packages **clusterCrit** and **NbClust**)
- max: **maximal** number of clusters: from 2 to 250 (YAML-500), from 2 to 100 (YAML-1140)

Hence, the *Vali-PP* process encompasses $3 \times 4 \times 3 \times 12 \times 249 = 107.568$ different combinations (*Vali-PP*-configurations) in the maximum case, each of them needing to be evaluated.

The new dimension

By "configuration of meta information" we mean a weighted combination of data fields which are extracted from the YAML meta information and which are used in the subsequent validation process. In the course of our study we propose three sample configurations, two extreme cases and one of our own choice. The "configurations of meta information" can be easily extracted by means of the packages **yamldebugger** and **TManalyzerQ**. Listing 3.10 demonstrates the use of the both packages. The function `yaml.list.extract` (from **TManalyzerQ**) relies on the function `yaml.getQField` (from **yamldebugger**).

1. **Configuration I**: Text documents are represented by descriptions and keywords data fields only, this case is thus minimum that could be reasonable for text mining and the most relevant as well.
2. **Configuration II**: Text documents are represented by all (not technical) data fields of the meta info, and this case is obviously the maximum one, including though a considerable amount of potentially uninformative and unreliable content.
3. **Configuration III**: Text documents are represented by weighted combinations of the most substantial fields (concerning the amount of words). Weights chosen were (6, 10, 3, 4, 5, 4) for the fields: "description", "keywords", "see also", "author", "datafile" and "example" correspondingly. The logic behind such a choice is that "description" and especially "keywords" reflect the most profound essence of the text and must thus make the greatest contribution. The next most important category is "datafile", since several QLs related to the same observation data set should be more or less close to each other. Less relevant are "author" and "example", because the author can also submit QLs from a completely different area and "example" often just duplicates the "description" content. Then at the end comes "see also" with only names of the similar documents in it.

```
library(yamldebugger)
library(TManalyzerQ)
(obj.names = load("yaml_list_full_20161122.RData", .GlobalEnv))  # 1140 Docs

help(yaml.getQField)
yaml.getQField(yaml_list[[1]], "d")

# Meta Configuration I
 t_vec = yaml.list.extract(yaml_list, weight = c(q=1, d=1, k=1, p=1))
# Meta Configuration II
 t_vec = yaml.list.extract(yaml_list)
# Meta Configuration III
 t_vec = yaml.list.extract(yaml_list, weight = c(d=6, k=10, sa=3, a=4, df=5, e=4))

str(t_vec)
> List of 2
> $ t_vec : chr [1:1140] " MSMLLN Plots the points showing law of large numbers." ...
> $ q_names: chr [1:1140] "MSMLLN" "MSM_LIL" "MSM_VaRandES" "MSMasprob" ...
```

Listing 3.10: Extraction of Meta Configurations via **yamldebugger** and **TManalyzerQ**

3.7.2 Experimental Procedure

For the clustering validation part of the *Vali-PP*, two R packages were used: **clusterCrit** and **NbClust**. By means of the first library we calculated all 12 quality indices (measures) as described before. From the second package those eight measures were taken, which intersect with the measures from **clusterCrit**, namely: Ball-Hall, C-Index, Calinski-Harabasz, Davies-Bouldin, Dunn, McClain-Rao, Ratkowsky-Lance and Silhouette. We had several reasons to include another library into our study: 1) to compare results for the same validation method from different packages; 2) to recalculate results for Silhouette index which were partially corrupted (and obviously useless) in the **clusterCrit** implementation; 3) to test whether those measures in **clusterCrit** that did not provide clear interpretation results could be easier interpreted using the other package. Within the package **clusterCrit** we compared the methods k-means, k-medoids and HC (*average* agglomeration), whereas from **NbClust** HC (*ward.D* and *ward.D2* agglomeration) and k-means were selected.

Two data sets were examined. The first one, YAML-500, was extracted from the YAML meta information of 500 QLs on December 26, 2015, see Section 3.7.3. The second one, namely YAML-

1140, was collected from 1140 QLs on November 22, 2016 and used in Section 3.7.4. The choice of 250 as maximum number of clusters for YAML-500 is justified as in Section "**3 Models, 3 Methods, 3 Measures**" in Borke and Härdle (2016). For YAML-1140 the cluster range within the interval $\{2, \ldots, 100\}$ was selected. Since the comprehensive $M^4_{d_1,d_2,d_3,d_4,250}$ analysis of the YAML-500 data has revealed the general properties of the *Vali-PP*-configurations, it was sufficient for the second stage to concentrate on the practically relevant cluster sizes, see last paragraph in Section 3.7.4.

3.7.3 Validation Pipeline Results

Index	Best model	Best clustering method	Best conf.
clusterCrit			
Ball-Hall (EI)	TT[13]/ LSA25[14]	HC	II
C-Index (EI)	TT/LSA25	HC	III
Calinski-Harabasz (EI)	LSA25	k-medoids[15]	II
McClain-Rao (EI)	LSA25	k-means/k-medoids[16]	III
Ratkowsky-Lance (EI)	TT/LSA25	all	I/II
Trace-W (EI)	LSA/LSA25	HC	II/III
Wemmert-Gancarski (EI)	LSA25	k-mean/k-medoids[17]	III
Davies-Bouldin (HI)	LSA25	k-medoids[18]	II
Ray-Turi(HI)	LSA25	HC	III
Xie-Beni (HI)	BVSM[19]	HC	II
Dunn (HI)	BVSM[20]	HC	II
NbClust			
Ball-Hall (EI)	TT[21]/ LSA25[22]	all	I/II/III
C-Index (HI)	LSA25	Ward D	II/III
Calinski-Harabasz (EI)	LSA25	Ward D/ Ward D2	II
McClain-Rao (EI)	LSA25	Ward D2	III
Ratkowsky-Lance (EI)	TT/LSA25	all	I/II/III
Davies-Bouldin (EI)	LSA25	k-means	I/II/III
Silhouette (EI)	LSA25	Ward D/ Ward D2	III
Dunn (HI)	BVSM[23]	Ward D2	II

Table 3.11: Main results of YAML-500 data for selected **clusterCrit** and **NbClust** quality indices

Table 3.11 shows the optimal TM model, clustering method and meta configuration ($d_1 \times d_2 \times d_3$) for each index, see Section 3.7.1 for the notation definitions. The last two dimensions/wheels of the *Vali-PP* (the appropriate index d_4 and the optimal number of clusters in the range

[13] TT model is the best wrt. optimal cluster number selection
[14] LSA25 model is the best wrt. global behavior of the curves
[15] k-medoids method is better, but HC is comparable from a certain cluster number size
[16] HC is also comparable in LSA25, which is the best model for this index
[17] HC is here comparable wrt. global behavior of the curves
[18] HC shows more stable behavior in all the models
[19] all models are very close in HC, in particular LSA and BVSM
[20] all models, in particular BVSM and LSA, are relatively close and are also in a quite small interval, regarding that the value range of this index is $[0, +\infty)$
[21] TT model is the best wrt. optimal cluster number selection
[22] LSA25 model is the best wrt. global behavior of the curves
[23] all models, in particular BVSM and LSA, are relatively close and are also in a quite small interval, regarding that the value range of this index is $[0, +\infty)$

$2, \ldots, max$) are to be considered in each particular case separately. However, some index results do not allow an unambiguous decision: Davies-Bouldin, Ray-Turi, Xie-Beni and Dunn. In these cases, necessary remarks are provided: **EI** - easy to interpret, **HI** - hard to interpret. Concerning the main results, the following (partly abbreviated) notations were used: TT for GVSM(TT), LSA25 for LSA space reduced to 25 dimensions, HC for *average* agglomeration, Ward D for HC (*ward.D* agglomeration) and Ward D2 for HC (*ward.D2* agglomeration).

The results and conclusions in Table 3.11 were made based on *Vali-PP* plots of YAML-500 with 4 curves (a curve for each vector space model/configuration d_2, in the following also referred to as *Vali-PP* graph) showing the value of a given validation index (Y axis) for each number of clusters from 2 to 250 (X axis). These plots with *Vali-PP* graphs were created for each combination of $d_1 \times d_3$. Altogether, we had 9 ($|d_1 \times d_3|$) *Vali-PP* plots for each index d_4, which resulted in $9*11 = 99$ *Vali-PP* plots for the package **clusterCrit** (11 indices, Silhouette index was skipped) and further $9*8 = 72$ *Vali-PP* plots for the package **NbClust** (8 indices). All validation results, *Vali-PP* plots and the corresponding Quantlets are available at Clustering_Validation_Pipeline.

The overall conclusion can be drawn that, when considered individually, the advantage of LSA or HC is sometimes not obvious, but their combination is mostly not worse than any other combination $d_2 \times d_3$ of a TM model and a clustering method. As for the meta information dimension d_1, no clear distinction can be made, whether configuration II or III is better. In any case, both of them outperform (or are not worse than) configuration I, which includes minimal information.

Concerning the results of the same quality indices from different packages (**NbClust** vs. **clusterCrit**), the best TM model is almost always the same: LSA25 (except from the Dunn index). The indicated optimal clustering method cannot be directly compared between the both packages, as only the k-means method is present as the "common element" in both cases. For the first stage of our validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$ it was sufficient to identify possible optimal combinations $d_1 \times d_2 \times d_3$ within each given package, looking at them separately from different angles: quality indices d_4, number of clusters $(2, \ldots, max)$ and various HC agglomeration methods.

3.7.4 Smart-Vali-PP

The main disadvantage of the manual/visual inspection and summarization of the validation results in Section 3.7.3 is the tedious and sometimes rather vague analysis of the *Vali-PP* plots for any given validation index. As discussed before, there are 3×3 *Vali-PP* plots, each of them containing 4 Vali-PP graphs. Furthermore, as shown in Table 3.11, it is usually difficult or even impossible to characterize an optimal *Vali-PP*-configuration for a given index d_4 in an unambiguous way, i.e. to say there is a particular optimal $d_1 \times d_2 \times d_3$ combination (meta/-model/method) for all considered cluster sizes $\{2, \ldots, max\}$. For that reason I have developed an improvised extension of the Validation Pipeline software infrastructure in order to facilitate the validation inspection process. This set of functions will be referred to as "Smart-Vali-PP" in the following. *Smart-Vali-PP* is a component of the overall *Vali-PP* infrastructure, which is displayed in Figure 3.7. In the near future, this code collection could be published as a separate R package.

The *Smart-Vali-PP* framework produces two essential outputs. One kind are the so-called "Smart-Vali-plots", which are generated using the *Vali-PP* results stored in a 5-dimensional array. For each index the aforementioned 9 plots ($d_1 \times d_3$ combination) are merged into a single plot. Together with four model configurations (d_2) every *Smart-Vali-plot* embeds 36 ($|d_1 \times d_2 \times d_3|$) different *Vali-PP* graphs (encoded in the standard TM colors) having the cluster size on the X axis. The black curve shows the "optimal function" (over all $d_1 \times d_2 \times d_3$ combinations) which can be achieved under the selected index. Two demonstrative *Smart-Vali-plots* are presented

in Figure 3.6. There we can clearly see that the LSA25 model is the optimal one both under C-Index and Silhouette index.

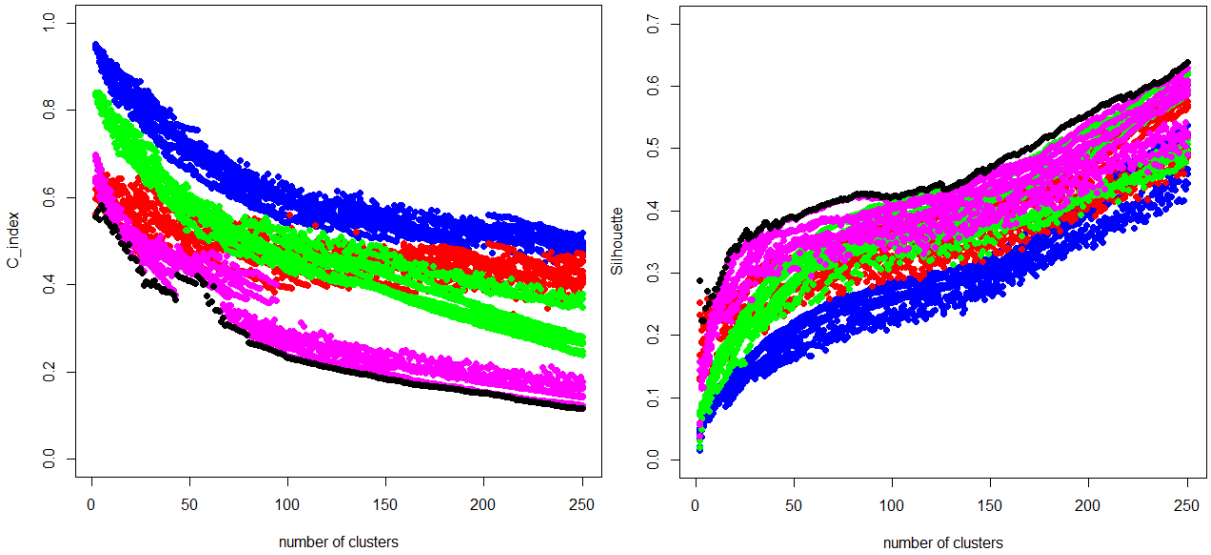


Figure 3.6: *Smart-Vali-plots* of YAML-500 for C-Index (left: to be minimized) and Silhouette index (right: to be maximized) from the package **NbClust**; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

Figure A.5 shows further *Smart-Vali-plots* created for the indices Ball-Hall, C-index and Ratkowsky-Lance. In this case, the current YAML-1140 data set was analyzed. The plots on the left side display only such *Vali-PP* graphs which intersect the "optimal function" at least at one point (one cluster size). The plots on the right side encompass all *Vali-PP* graphs, allowing the analysis of the overall shape and trends of all *Vali-PP*-configurations $d_1 \times d_2 \times d_3$. The *Smart-Vali-PP* function `plot.optimal.functions` from Listing A.5 carries out the *Smart-Vali-plots* creation.

The other kind of *Smart-Vali-PP* outputs are "Smart-Vali-tables" as shown in Table 3.12. For each index, the triple $d_1 \times d_2 \times d_3$ (representing a particular *Vali-PP* graph) is aggregated and sorted according to the cluster sizes where this triple coincides with the "optimal function". The additional columns "Frequency", "Cumulative relative proportion" and "Relative proportion" allow to identify and quantify the index optimality for a given *Vali-PP*-configuration $d_1 \times d_2 \times d_3$ along the dimension "cluster size". The function `optimal_share.prettify` from Listing A.5 carries out the *Smart-Vali-tables* creation.

Both the *Smart-Vali-plots* and the *Smart-Vali-tables* are different representations based on the data structure `optimal_share_index`, see Listing A.5. Looking for instance at the C-Index in Figure A.5 and Table 3.12, we can easily conclude that the GVSM(TT) Vali-PP graphs are mostly in proximity to the "optimal function". Additionally, there are one LSA50 Vali-PP graph and two BVSM Vali-PP graphs which intersect the goal function at least at one cluster size. By means of the *Smart-Vali-table* we can infer that the GVSM(TT) ($d_2 = 2$) reaches the optimal function at 92 cluster sizes of 99. LSA50 ($d_2 = 4$) accomplishes that at 5 cluster sizes and BVSM ($d_2 = 1$) 2 times. Further we can see that the *Vali-PP*-configuration $(d_1, d_2, d_3) = (3, 2, 3)$ appears at 50 of 99 cluster sizes and hence clearly dominates. The next best configurations are $(d_1, d_2, d_3) = (3, 2, 2)$ and $(d_1, d_2, d_3) = (3, 2, 1)$, what means that GVSM(TT) and meta configuration III is best in all three cases, representing 80% of all considered cluster sizes. Concerning the method (d_3) in these three cases, we have the optimality order: 3, 2, 1, i.e. HC, k-medoids, k-means. All dimension values in d_1, d_2, d_3 are enumerated in the same order as listed in Section 3.7.1, e.g. $d_2 = 2$ means GVSM(TT) and $d_1 = 1$ means meta Configuration I.

Conf. d_1	Method d_3	Model d_2	Freq.	Cum. rel. prop.	Rel. prop.
C-Index					
3	3	2	50	0.50	0.50
3	2	2	15	0.66	0.15
3	1	2	14	0.80	0.14
1	1	2	10	0.90	0.10
2	3	4	5	0.95	0.05
1	2	2	3	0.98	0.03
2	3	1	1	0.99	0.01
3	3	1	1	1.00	0.01
Wemmert-Gancarski					
2	3	4	68	0.69	0.69
3	3	4	20	0.89	0.20
3	2	2	5	0.94	0.05
3	3	2	3	0.97	0.03
3	2	4	2	0.99	0.02
2	2	4	1	1.00	0.01
Ray-Turi					
3	3	4	38	0.38	0.38
1	3	4	20	0.59	0.20
3	3	2	18	0.77	0.18
2	3	4	17	0.94	0.17
2	3	2	4	0.98	0.04
3	2	2	1	0.99	0.01
1	3	2	1	1.00	0.01

Table 3.12: *Smart-Vali-tables* of YAML-1140 for the indices: C-Index, Wemmert-Gancarski, Ray-Turi from the **clusterCrit** package

In a similar manner it can be concluded from the *Smart-Vali-tables* that LSA50 and HC is optimal in 89% of the cluster sizes under the Wemmert-Gancarski index, and LSA50 and HC is also optimal in 75% of the cluster sizes wrt. the Ray-Turi index. The meta configuration co-occurrence can be inferred from the *Smart-Vali-tables* in an analogous manner. In other words, *Smart-Vali-tables* permit statistical analysis of the co-occurrence distribution of the optimal *Vali-PP*-configurations. Concerning the YAML-1140 data set, only cluster sizes from 2 up to 100 were analyzed because the current QuantNetXploRer implementation requires cluster sizes in a range from 16 to 64. Thanks to the *Smart-Vali-plots*, one can easily see that it needs an initial period of cluster sizes (about 20) until the index functions start to consolidate their decreasing or increasing trend.

3.7.5 Smart-Vali-PP summary

The YAML-1140 *Smart-Vali-PP* results for all 12 quality indices (measures) are summarized in Section A.10. The optimal *Vali-PP*-configurations of the examined quality indices can be classified into three different groups, see Table 3.13. Six of twelve quality indices are assigned to the group *GVSM HC*, encompassing all GVSM(TT) and LSA representations and the clustering method HC. The relative percentage values in Table 3.13 indicate the proportion of the cluster sizes for which the given classification applies. For instance, the entry Ray-Turi $\times$ GVSM HC : 0.99 means that the co-occurrence of a GVSM representation (GVSM(TT) or

LSA) and the HC method is optimal in 99% of the cluster sizes under the Ray-Turi index. The classification *SMART GVSM* comprises all GVSM representations (GVSM(TT) or LSA) together with the (smart) meta configuration III, see also Section 3.7.1. There are two quality indices, namely Ball-Hall and C-index, which are included in both classifications (GVSM HC and SMART GVSM). The group assignments are not unigue, thereby allowing to choose the appropriate characterization of the *Vali-PP*-configurations, depending on the requirements of the analysis. The relative percentage values serve as reference quantities. The last group is *MAX HC* containing the (maximum) meta configuration II and the clustering method HC as the optimal co-occurrence configuration.

Clustering quality index	GVSM HC	SMART GVSM	MAX HC
Ray-Turi	0.99		
Silhouette[24]	0.99		
Wemmert-Gancarski	0.92		
Davies-Bouldin	0.66		
Ball-Hall	0.58	1.0	
C-index	0.56	0.8	
Ratkowsky-Lance		1.0	
Calinski-Harabasz		1.0	
Trace-W		1.0	
McClain-Rao		0.96	
Xie-Beni			1.0
Dunn			0.97

Table 3.13: YAML-1140 *Smart-Vali-PP* results for all 12 quality indices

The YAML-1140 *Smart-Vali-PP* results can be summarized as follows: 6 of 12 quality indices belong to the group GVSM HC, 6 of 12 indices belong to the group SMART GVSM, whereby the both groups share two common indices. The remaining two indices are characterized by the numerically smallest group MAX HC. The last group is the only group of indices where the GVSM representations are not dominant. However, the maximum metadata magnitude assumes the role of a dominant driving factor. It is also worth mentioning that the groups SMART GVSM and MAX HC have a very high cluster size coverage (except the C-index more than 95%).

3.7.6 Vali-PP software and hardware infrastructure

In Figure 3.7 all software components are displayed which were integrated and developed in the course of the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$. They consist of various R functions, our R packages (in orange-blue), external R packages (in blue), and the *Smart-Vali-PP* component.

Particularly noteworthy features within the Vali-PP software infrastructure are the following. The object `optimal_share_index` in Listing A.5 is basically a 3-dimensional table counting the number of intersections between every possible *Vali-PP*-configuration $d_1 \times d_2 \times d_3$ and the "optimal function". Due to the smart function `multi.which` from Listing A.4, which was created by Mark van der Loo[25], all 3 *Smart-Vali-PP* functions `optimal_share.for.index`, `optimal_share.prettify` and `plot.optimal.functions` could be realized in a very compact and elegant way.

[24] the Silhouette index results were taken from the YAML-500 **NbClust** package validation, see Section 3.7.2
[25] `https://github.com/markvanderloo`

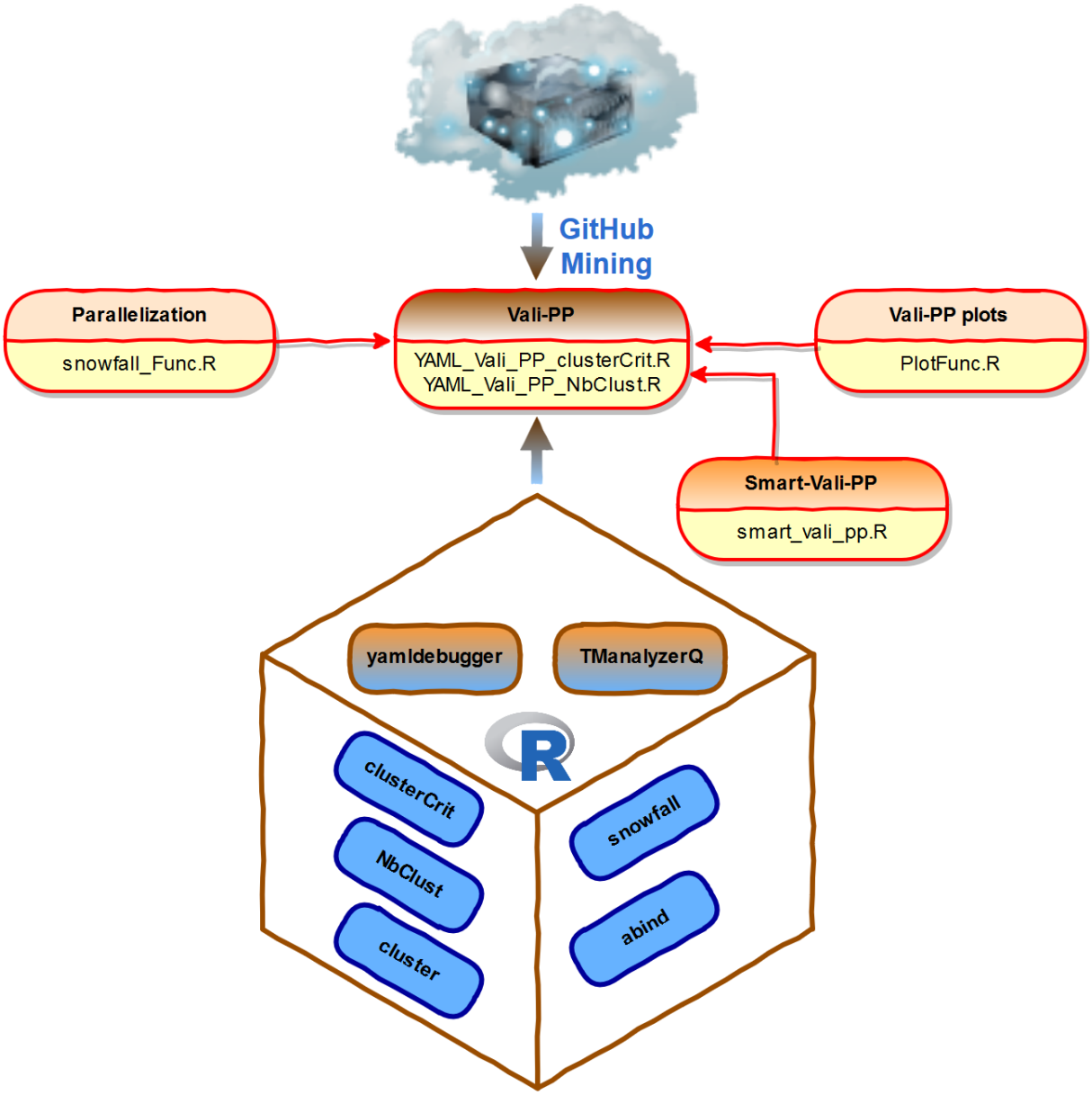


Figure 3.7: Software infrastructure of the Validation Pipeline components; blue: external R packages, orange-blue: our R packages; orange: our R functions;

Since the *Vali-PP* calculations were time consuming, the whole process was parallelized over 24 or 32 CPU cores, depending on the used calculation server (Research Data Center, `https://rdc.hu-berlin.de`). The dimension "number of clusters" is the ideal quantity for massive parallelization. This allowed to complete the validation benchmark for the YAML-500 data within one day for every package. In the case of **NbClust**, it took sometimes more than four hours to compute the results for a single index. The execution of **clusterCrit** ran faster, but still took approximately 10 hours for all 12 indices. The obvious reason for the better performance of **clusterCrit** is its C and Fortran 95 optimization[26].

[26] `https://github.com/cran/clusterCrit/tree/master/src`

Concerning YAML-1140, the *Vali-PP* benchmark was performed only for the **clusterCrit** package and cluster sizes up to 100. As discussed before, **clusterCrit** is better optimized, has a broader spectrum of quality indices and the upper limit of 100 clusters satisfies the practical needs. According to the results in Section "3.3 Benchmark" in Desgraupes (2013), the `intCriteria` function is quite efficient because the code is optimized to avoid duplicate calculations and to reuse values already computed for other indices. The function `clusterCritParallel` within the *Vali-PP* infrastructure was therefore adjusted. The vector of all selected indices is passed to the `intCriteria` function at once. Together with some other improvements like the calculation of the distance matrix in advance (for the `pam` and `hclust` clustering functions), the following calculation times were measured, see Table 3.14. For better handling, the benchmark was executed for each meta configuration separately, the other four dimensions (*Vali-PP*-configurations) were processed in one pass. The full calculation for all meta configurations, and with it the entire *Vali-PP* benchmark, took around 2 hours and 20 minutes.

Meta Configuration	time in seconds	physical/logical cores
I	2410	24/24
II	2664	24/24
III	3364	12/24

Table 3.14: *Vali-PP* calculation time for YAML-1140 data, grouped by meta configurations

As the main objective of this section, we introduced and examined the Validation Pipeline, a functional multi-staged instrument for clustering analysis, allowing multivariate statistical analysis of the distribution of driving factors (*Vali-PP*-configurations).

3.8 Discussion

3.8.1 Conclusion

The new package **rgithubQ** enables a GitHub wide search for code and repositories using the GitHub Search API and allows to implement different parsers for data extraction from various software repositories retrieving smart data out of the raw text collection. Performing similar as Google, it is designed to find results that best meet the personal needs and which are ranked by best match, as indicated by the score field for each item returned. The QuantNet@GitHub statistics within **rgithubQ** can be retrieved in real time. Due to some lightweight parsers of the package, basic mining tasks on QuantNet can be performed directly in the R console by use of **rgithubQ** without further QuantNet Mining infrastructure R components.

The QuantNet Style Guide and the **yamldebugger** package allow a standardized audit and validation of YAML annotated software repositories. First, the **yamldebugger** checks the Quantlet repository structure, the validity of the YAML meta information and the completeness of the mandatory data fields according to the Style Guide. Second, it helps to analyze and unify the different YAML data fields, which are subject to varying spelling and notations. A meaningful and reasonable calibration of the matching list of the YAML data fields within the **yamldebugger** is crucial for further extraction of smart data within the TM and cluster validation steps.

The presented Google Analytics driven QuantNet Test Collection obtained from Google's metrics was used to evaluate and calibrate the IR performance. Three common text mining (TM) models were examined by means of the novel **TManalyzerQ** package. By generating three performance measurement matrices: the number of retrieved documents, the number of retrieved and relevant documents (true positives), and the precision value for each query $\times$ TM model

combination, the **TManalyzerQ** constitutes a convenient tool for IR design and performance analysis. In this way, we can conclude that the number of true positives for all considered single term queries is maximal in the LSA50 model.

Further, we introduced the **Validation Pipeline** (*Vali-PP*), a functional multi-staged instrument for clustering analysis, providing multivariate statistical analysis of the co-occurrence distribution of driving factors of the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$. The *Smart-Vali-PP* framework, being a component of the overall *Vali-PP* infrastructure, allows to identify and quantify the clustering index optimality for a given *Vali-PP*-configuration. It can be applied to each of the 27 internal quality indices offered by the package **clusterCrit**. Considering the examined quality indices, the optimal *Vali-PP*-configurations can be classified into three different categories: GVSM HC, SMART GVSM and MAX HC, which demonstrates that generalized vector space models (GVSM including LSA), accurate and comprehensive metadata (SMART, MAX) and hierarchical clustering maximize the clustering quality. We can also infer that the co-occurrence frequency of the meta configurations II (maximum) and III (smart) is high (and even reached 100% for the indices: Wemmert-Gancarski, Ball-Hall, Ratkowsky-Lance, Calinski-Harabasz, Trace-W, Xie-Beni and Dunn) among the optimal *Vali-PP*-configurations, which underlines the importance of metadata for the clustering quality. It is also worth mentioning that the categories SMART GVSM and MAX HC reveal a very high cluster size coverage. Within the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$, the packages **yamldebugger** and **TManalyzerQ** were used to produce the meta information configurations and TM models.

3.8.2 Future Perspectives

The **TManalyzerQ** can be directly connected to the parser layer of the "GitHub API based QuantNet Mining infrastructure" and run as an R based search engine with three implicit TM models. Hence, the packages **rgithubQ**, **yamldebugger** and **TManalyzerQ** are the necessary components for a self-contained GitHub mining engine in R. Due to the general approach, any set of text documents can serve as an object of text analysis. The **yamldebugger** acts as a smart data extraction layer and can be omitted or replaced by another text preprocessing component if another type of data is involved.

The new R packages presented in this paper build the integral components of the "GitHub API based QuantNet Mining infrastructure in R". The remaining components, such as the parser, clustering and D3 export layers, are available in experimental and working state. Very soon they will be implemented in a new R package, together with our research findings. Thus, the TM pipeline introduced in the beginning of this paper will be available in form of a package under the name "**tmPipelineQ**" and QuantNet's search engine called QuantNetXploRer will be finalized. The GitHub API driven QuantNetXploRer can be already found and used under `http://www.quantlet.de`.

> The psychological profiling [of a programmer] is mostly the ability to shift levels of abstraction, from low level to high level. To see something in the small and to see something in the large.
>
> *An interview with Donald Knuth. Dr. Dobb's Journal (April 1996)*

4 RiskAnalytics: an R package for real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods

4.1 Software implementation in R

Koenker and Mizera (2014) survey some recent developments of convex optimization and describe some implementations of these methods in R. Quadratic programming (QP), as part of convex optimization, involves the minimization of a positive semi-definite quadratic objective function subject to polyhedral constraints. There are many applications of QP in statistics, typically involving Gaussian likelihoods constrained by some form of linear inequalities. Shape constrained regression examples have gained recent attention, and the introduction of sparse regularization methods like lasso, has greatly stimulated interest in computational methods for such problems. One of the most familiar statistical QP applications in recent times has been the lasso estimator of Tibshirani (1996).

Standard quantile regression (QR) models can be estimated with the `rq()` function of the **quantreg** package (Koenker, 2016). However, software implementations for computing solution paths of lasso penalized QR are rare. **hqreg** (Yi, 2016) is such an example. This R package is relatively new (it was published for the first time on 21 June 2015), its version history is trackable on GitHub[1]. The main advantage is its C optimization[2]. Yi and Huang (2015) demonstrate both the convergence properties of the proposed algorithm and the numerical experiments, showing that their package implementation is very efficient and scalable to ultra-high dimensions.

Another available R implementation is the supplementary code of Li and Zhu (2008), which can be found in QuantNet: QFRM_lambda_series. At the time of the early stage development of the FRM project (Yu et al., 2017), only the latter code was known and available. Therefore, the current lasso penalized QR implementation of FRM relies on the idea of Li and Zhu (2008). In the following, numerical experiments and benchmarks will be provided in order to evaluate the speed and efficiency of the current FRM version.

4.2 CRAN Mining

Listing 4.1 demonstrates the "GitHub Mining" capabilities of the **rgithubQ** package. The search query with the search string "quantile lasso regression" finds all locations on GitHub of the CRAN typical *DESCRIPTION file* containing this special term. Five matches were retrieved sorted by the score field. The R package **hqreg** (see Section 4.1) is found twice. The first result is the developer version hosted on the account of the author[3]. The second result is located in the CRAN organization[4]. All output details like search score, submission date, version, authors

[1] `https://github.com/cran/hqreg/commits/master`
[2] `https://github.com/cran/hqreg/blob/master/src/hqreg.c`
[3] `https://github.com/CY-dev/hqreg`
[4] `https://github.com/cran`

etc. are listed in Tables 4.1 and A.1.

```
library(rgithubQ)
# GitHub's user authorization
ctx = interactive.login("client_id", "client_secret")

r_pack_search = 'Package Title Version Description Author
                 filename:"DESCRIPTION" path:"/"'
spec_search_term = "quantile lasso regression"
sr = search.code(paste(spec_search_term, r_pack_search), per_page = 10)
sr$content$total_count

dcf_top = dcf.parser.light(sr, print_item = FALSE)
( p_name    = sapply( dcf_top, function(dcf){ dcf$Package } ) )
( p_title   = sapply( dcf_top, function(dcf){ dcf$Title } ) )
( p_date    = sapply( dcf_top, function(dcf){ dcf$Date } ) )
( p_version = sapply( dcf_top, function(dcf){ dcf$Version } ) )
( p_author  = sapply( dcf_top, function(dcf){ dcf$Author } ) )
( p_path    = sapply( sr$content$items, function(item){ item$repository$full_name } ) )
( p_scores  = sapply( sr$content$items, function(item){ item$score} ) )

( name_path_title_scores = data.frame(name = p_name, repo.path = p_path,
                          score = round(p_scores, 2), package.title = p_title) )
( path_date_version_author = data.frame(repo.path = p_path, date = p_date,
                            version = p_version, package.author = p_author ) )
```

Listing 4.1: Search Code for CRAN packages via the **rgithubQ** package

	name	repo.path	score	package.title
1	hqreg	CY-dev/hqreg	6.23	Regularization Paths for Lasso or Elastic...
2	hqreg	cran/hqreg	5.94	Regularization Paths for Lasso or Elastic...
3	cqrReg	cran/cqrReg	5.86	Quantile, Composite Quantile Regression...
4	rqPen	bssherwood/rqpen	5.82	Penalized Quantile Regression
5	rqPen	cran/rqPen	5.45	Penalized Quantile Regression

Table 4.1: All R packages on GitHub dealing with "quantile lasso regression", extracted via **rgithubQ**

4.3 RiskAnalytics package

In order to integrate and facilitate the research, calculation and analysis methods around the FRM project (Yu et al., 2017), the R package **RiskAnalytics** (Borke, 2017b) has been developed. Its main goal is to provide data processing and parallelized quantile lasso regression methods for risk analysis based on NASDAQ data, Yahoo Finance data and the macro variables as described in the Data section in Yu et al. (2017). The derived "Risk Analytics" can help to forecast and evaluate the systemic risk for the corresponding markets.

As member of the Research Data Center (RDC) I was involved in the development of the FRM project from the very beginning, having the main tasks: automation of data collection, optimization and parallelization of code (lasso penalized QR), and data visualization (risk meter designs). Based on this experience, the functionality of the **RiskAnalytics** package is subdivided into 4 major software components:

1) data processing (*get_data.R*);
2) parallel computing (*parallel_calculation.R*);
3) QR methods (*qrL1.R*);
4) "Risk Analytics" (*analytics.R*);

Every software component contains several related functions. Their interaction is presented in Listings 4.2, 4.3, 4.4, 4.5 and 4.6.

4.3.1 RiskAnalytics package: data extraction and analysis part

```
#-----------------------------------------------
# Initialization
#-----------------------------------------------
library(snow)
library(RiskAnalytics)
work_dir = "c:/r/frm/2017"
max_companies = 100

#-----------------------------------------------
# Load data
#-----------------------------------------------
companylist = get.nasdaq.companies()

system.time( yahoo_data <- get.yahoo.data(companylist, max_comp_num = max_companies, from_date =
     "2006-12-29") )
# truncated output for illustration
[1] "97 : SBNY"
[1] "98 : ZION"
[1] "diff length : CIT"
[1] "diff length : APO"
[1] "99 : WRB"
[1] "100 : SEIC"
      User     System    Elapsed
      7.85       1.44      58.20

system.time( macro_data <- get.macro.data(from_date = "2006-12-28") )
      User     System    Elapsed
      1.05       0.06       5.13

final_data = combine.data(yahoo_data, macro_data, summary_dim = c(1:3, 102:107))
[1] "Dimension of the final data: 2534 * 107"
        Date         JPM         WFC       ^VIX     ^GSPC      IYR     3MTCM
1  03/01/2007 0.002290948 0.005049110 0.02353107 0.4414408 0.5978950 0.4904459
2  04/01/2007 0.002493227 0.001677339 0.03029449 0.4577132 0.6204483 0.5159236
3  05/01/2007 -0.008335091 -0.005602276 0.02282655 0.4695943 0.6035441 0.5222930
4  08/01/2007 0.003342404 -0.002812915 0.03170354 0.4337065 0.5632682 0.5286624
5  09/01/2007 -0.004179761 0.002532009 0.02973087 0.4744425 0.6035341 0.4840764

data.analytics(yahoo_data, macro_data)
# truncated output for illustration, correlation matrix of the macro var's
       ^VIX ^GSPC  IYR 3MTCM Yield Credit
^VIX   1.00 -0.14 -0.11 -0.06 0.26 0.55
^GSPC -0.14  1.00  0.81 -0.02 0.00  0.01
IYR   -0.11  0.81  1.00 -0.04 0.01  0.02
3MTCM -0.06 -0.02 -0.04  1.00 0.00  0.00
Yield  0.26  0.00  0.01  0.00 1.00  0.36
Credit 0.55  0.01  0.02  0.00 0.36  1.00
```

Listing 4.2: RiskAnalytics application example: data extraction and analysis part

Listing 4.2 demonstrates the data extraction and analysis part of the **RiskAnalytics**. The functions `get.nasdaq.companies`, `get.yahoo.data` and `get.macro.data` are responsible for real time processing of NASDAQ, Yahoo Finance and Federal Reserve Bank of St. Louis data. `get.nasdaq.companies` extracts the top NASDAQ companies (sorted by their market capital-

ization) from the web resource[5] by means of the package **RCurl** (Lang and the CRAN team, 2016). `get.yahoo.data` provides daily log returns of the selected NASDAQ companies by use of the package **quantmod** (Ryan, 2016). `get.macro.data`, in its turn, employs both approaches: Yahoo Finance via **quantmod** for the download of the VIX, GSPC (S&P500) and IYR (iShares Dow Jones US Real Estate) macro variables, and direct downloads of the other 3 macro variables from the corresponding web resources on `https://fred.stlouisfed.org/`.

The helping function `combine.data` combines all previously obtained time series in an appropriate time and date format. Additionally, the dimension and the preview of a sub sample of the resulting data frame object is displayed. The latter can be controlled by the parameter `summary_dim`, see also Listing 4.2. All aforementioned functions provide additional information and, where appropriate, graphical plots for better audit and validation checks of the extracted data, see e.g., Figures 4.1 and 4.2.

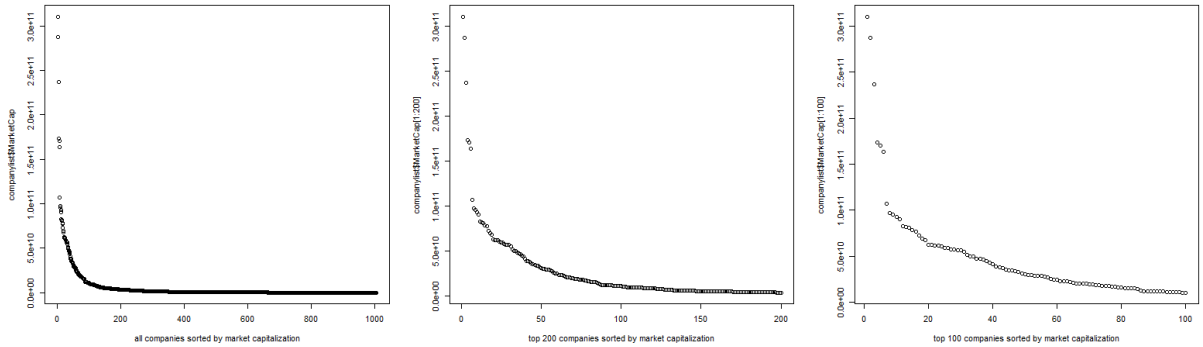


Figure 4.1: NASDAQ companies sorted by the market capitalization: all (left), top 200 (middle) and top 100 (right), produced via *get.nasdaq.companies*

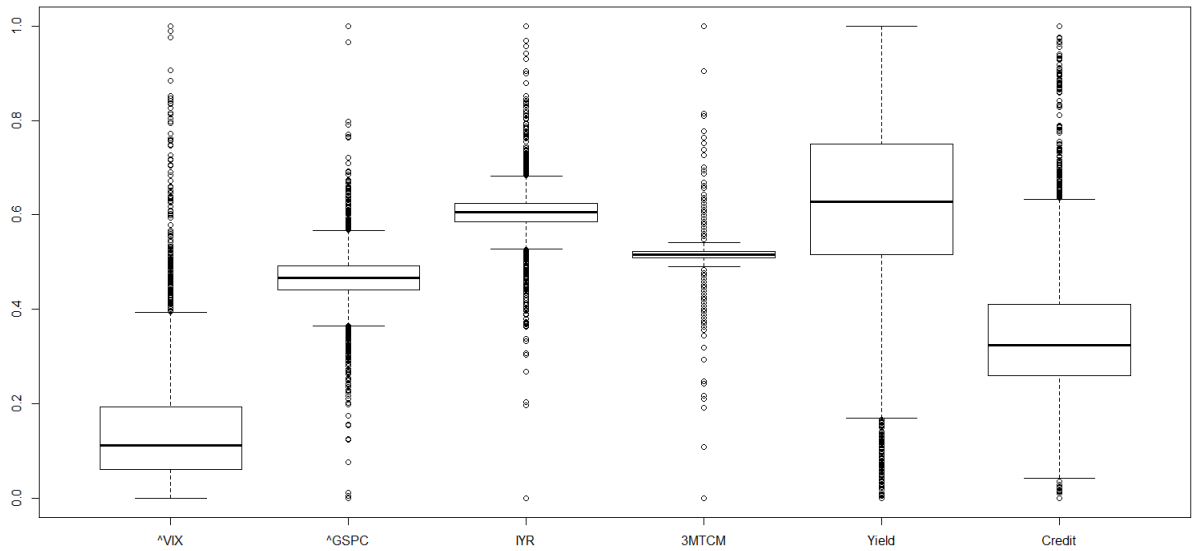


Figure 4.2: Box plots of macro variables produced via *get.macro.data*

The function `data.analytics` from Listing 4.2, which is actually a part of the "Risk Analytics" software component, provides descriptive statistics for both the NASDAQ companies and the macro variables. All statistical information vital for the subsequent QR methods is summarized in a brief overview. For instance, it becomes immediately obvious that the macro

[5] `http://www.nasdaq.com/screening/companies-by-industry.aspx?industry=Finance`

variables will be dominant regressors due to their larger Euclidean norms, compared to those of the NASDAQ companies (see the box plots in Figure 4.3). Together with the output in Figure 4.2 and 4.4, one can easily conclude that the macro variables VIX (1), "Yield spread" (3) and "Credit spread" (4) (see the Data section in Yu et al. (2017) for the enumeration assignment) will be "driving factors" in the QR process because of their high variances. Furthermore, `data.analytics` returns also the correlation matrix of all six macro variables, revealing that the aforementioned variables VIX, "Yield spread" and "Credit spread" have positive correlations among each other, see Listing 4.2. In the light of this technical analysis, it is hardly surprising that both the FRM and VIX time series reveal a similar behavior, see the "FRM versus VIX" section in Yu et al. (2017).

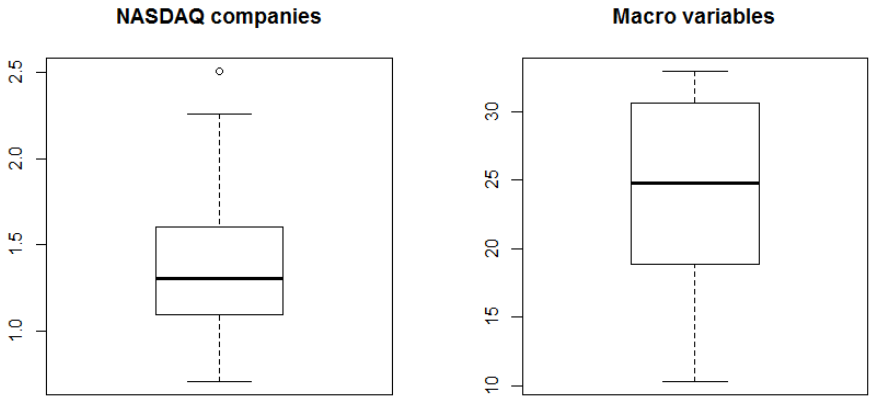


Figure 4.3: Box plots of the euclidean norms of the Yahoo Finance data/companies (left) and the macro variables (right), produced via *data.analytics*

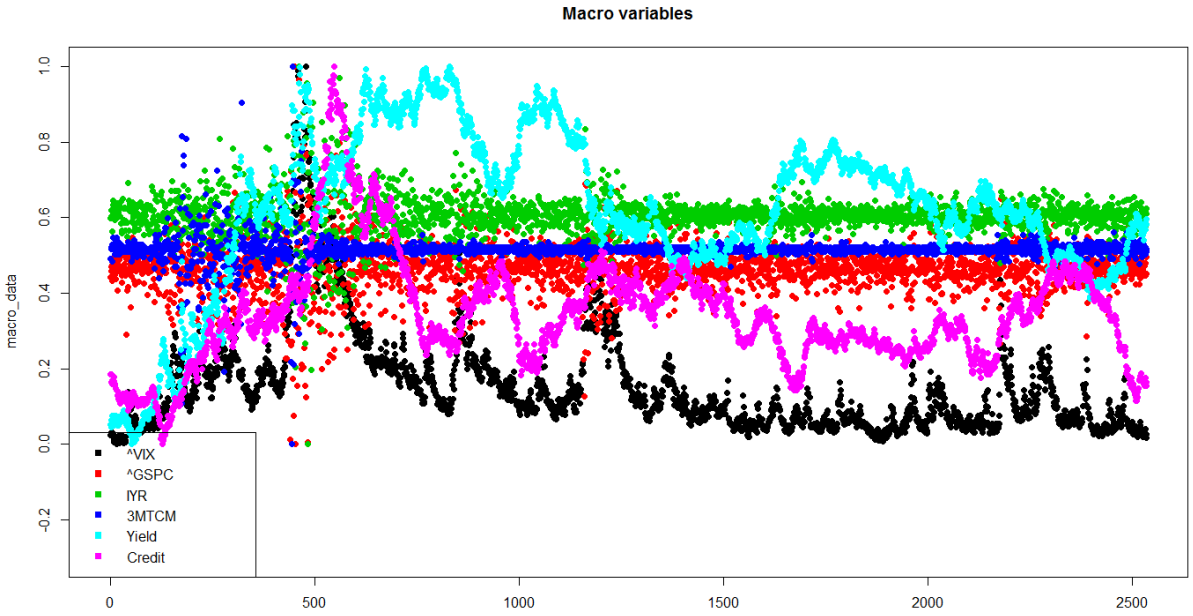


Figure 4.4: Plot of the macro variables, produced via *data.analytics*

According to Listing 4.2, the data processing component extracts all needed data in around one minute. Additionally, the `data.analytics` function provides statistical information for the further QR process. All obtained data are stored in the RAM, hence no further write or storage operations are required, and the real time data can be passed over to the next component: parallel computing.

4.3.2 RiskAnalytics package: parallel computing part

Listing 4.3 shows the execution and benchmark results of the parallel computing component of **RiskAnalytics**. Based on the packages **snow** (Tierney et al., 2016) and **snowfall** (Knaus, 2015) and the lasso penalized QR implementation of Li and Zhu (2008), the calculation of the QR method is performed for all moving windows and all NASDAQ companies. The most important parameters of the function `parallel.lasso.computation` are `max_companies`, `new_days`, `parallel_cpu`, `p`, `winsize` meaning: 1) number of desired NASDAQ companies, the parallelization is performed along this dimension; 2) number of desired moving windows within the total data observation time frame; 3) number of available CPU's for the parallel computing via **snowfall**; 4) desired quantile value for the QR method; and 5) the length of the moving window. Most of these parameters have default values as displayed at the beginning of Listing 4.3.

```
#-----------------------------------------------
# Calculate data
#-----------------------------------------------

# by default: new_days = 5, parallel_cpu = 4, p = 0.05, winsize = 60

# main calculation for the FRM visualization
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir, new_days
    = 2469, parallel_cpu = 32, winsize = 63)
# R Version: R version 3.3.2 (2016-10-31)
# snowfall 1.84-6.1 initialized (using snow 0.4-2): parallel execution on 32 CPUs.
# Stopping cluster
#    user  system elapsed
#    1.45    3.43 54895.78

# test benchmark for 200 working days, ca. 10 months
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir, new_days
    = 200, parallel_cpu = 32)
# Stopping cluster
#    user  system elapsed
#    0.14    0.14 4287.58

# test benchmark for 5 working days, 1 working week
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir, parallel_
    cpu = 32)
# Stopping cluster
#    user  system elapsed
#    0.17    0.14  115.89
```

Listing 4.3: RiskAnalytics application example: parallel computing part

For each company and each moving window the QR results are stored in the data structure `parResult`. The latter is basically a list with elements corresponding to the companies. Every list element j contains the lambda values (λ_j) and beta coefficients (β_j) from the QR procedure. For a given company j, the lambda values are a vector enumerated by the calculated days `new_days`, and the beta coefficients are a matrix, whose rows are the calculated days `new_days` and whose columns are the regressors/covariates.

According to Li and Zhu (2008), the computational complexity of the L_1-norm QR algorithm is $\mathcal{O}(p \min(n,p)^3)$, with n being the length of the moving window and p the number of covariates. The main calculation for the FRM visualization is performed with $n = 63$ and $p = 105$ (99 companies except the regressed one and 6 macro variables), see also Listing 4.3. In addition to the basic complexity, we have to deal with two further dimensions, i.e. n_c (`max_companies` or *number* of *companies*) and n_w (`new_days` or *number* of moving *windows*).

In summary, the `parallel.lasso.computation` function for the main calculation of the *FRM* lambda time series has a computational complexity of

$$\mathcal{O}(n_c n_w)\mathcal{O}(p\min(n,p)^3), \tag{4.1}$$

which results in approximately 6.5×10^{12} basic calculations, if we compute the *FRM* lambda for $n_w = 2469$ (around 10 years).

The time complexity benchmarks for four cases: $n_w = 2469$, $n_w = 200$, $n_w = 10$ and $n_w = 5$ are provided in Table 4.2, see Listing 4.3 for some examples. The tests were performed on a RDC Windows server with 16 physical and 32 logical cores and Intel Xeon CPU E5-2690 0 @ 2.90 GHz. In each case `max_companies` was equal to 100, `parallel_cpu` = 32, `p` = 0.05. The corresponding length of the moving window n (`winsize`) and the number of moving windows n_w (`new_days`) are given in the table columns.

n (window size)	n_w	time in seconds	time in minutes	time in hours
60	5	116	2	0.03
60	10	222	4	0.06
60	200	4288	71	1.19
63	2469	54896	915	15.25

Table 4.2: Time complexity benchmarks for *parallel.lasso.computation* of the **RiskAnalytics** package

As can be expected from Formula 4.1, the running time of `parallel.lasso.computation` scales in proportion to n_w. For a better comparison, the same time measurements are displayed in seconds, minutes and hours, respectively. As main results of the time complexity benchmarks, we can conclude that:

I) The lasso penalized QR implementation in FRM can be performed within 2 minutes for the calculation of 5 working days and within 15 hours for a time period of 10 years, which shows that the QR calculation is feasible on a contemporary computer with 16 physical cores.

II) For the increase of the speed, only the physical CPU cores are relevant, what means that the calculations can be performed on a usual home PC with 4 CPU cores, like for example Intel Core i5-2500 with 4 physical cores. In this case, the time demand must be multiplied by factor of 4 (16 cores ÷ 4 cores).

III) The memory demand for the storage of all necessary data and calculation results is very modest and is mainly dictated by the dimensions of the data matrices and frames and the data structure `parResult`. Saved as files, the data object `final_data` from Listing 4.2 and `parResult` from Listing 4.3 require around 2 MByte and 60 MByte, respectively.

The `parallel.lasso.computation` function accepts some additional optional parameters for minor validation outputs and allowing to save the calculation results as file outputs. By default, the parallel computing component operates as an "in-memory application" without requiring any disk Input/Output operations.

4.3.3 RiskAnalytics package: QR.analytics part

The code examples in Listing 4.4 demonstrate the *QR.analytics* part of **RiskAnalytics**, the former being a subset of the "Risk Analytics" software component. QR.analytics comprises 3 functions: `QR.regressors.stats`, `QR.beta.stats` and `QR.variance.vs.beta`. The output `parResult` from the parallel computing part serves as an "object of investigation".

The functions `QR.regressors.stats` and `QR.beta.stats` analyze the structure of the beta coefficients from the QR process. `QR.regressors.stats` provides the frequency of the covariates

for a given percentage threshold `sel_threshold` and the filter value `min_regressed_comp`. For instance, for a given `sel_threshold` = 0.55 and `min_regressed_comp` = 10, we see in the first part of Listing 4.4 that only the following covariates: 88, 101, 102, 103, 105, 106 (88 is the number of a NASDAQ company, numbers higher than 100 are macro variables) have non-zero beta coefficients in the QR of a company. Additionally, we have the restrictions that the filtered and displayed covariates are active regressors in at least 55% of all moving windows (n_w) for at least 10 companies. For `sel_threshold` = 0.55 and `min_regressed_comp` = 10 we can conclude that the company with the number 88 is an active regressor for some 11 companies, being present in at least 55% of all moving windows for each of those 11 companies. The macro variable with the number 101 (VIX), on the other hand, is an active regressor with non-zero beta's in all 100 NASDAQ companies, being present in at least 55% of all moving windows for each of them. The second most influential regressor is the macro variable with the number 102 (S&P500), it is an active regressor for 99 NASDAQ companies (in at least 55% of all moving windows).

```
#-----------------------------------------------
# QR.analytics: QR.regressors.stats
#-----------------------------------------------
sapply( c(0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.72), function(x) { QR.regressors.stats(parResult,
    sel_threshold = x, min_regressed_comp = 10 )$R_tab_min_regressed_comp })

[[1]]
 15  22  24  31  55  56  58  63  67  81  88  90 101 102 103 105 106
 31  46  51  11  74  21  55  15  10  11  79  73 100 100 100 100 100

[[2]]
 22  24  55  58  88  90 101 102 103 105 106
 13  15  23  21  43  37 100 100 93 100 98

[[3]]                       [[4]]
 88 101 102 103 105 106     101 102 103 105 106
 11 100 99  67  86  72       98  99  15  30  16

[[5]]         [[6]]        [[7]]
101 102       101 102      102
 76  87        14  38       16
#-----------------------------------------------
# QR.analytics: QR.beta.stats
#-----------------------------------------------
ave_beta_share = QR.beta.stats(parResult)
which(ave_beta_share > 0.5)
[1] 101 102 103 105 106
which(ave_beta_share > 0.666)
[1] 101 102
#-----------------------------------------------
# QR.analytics: QR.variance.vs.beta
#-----------------------------------------------
variance_vs_beta = QR.variance.vs.beta(final_data, ave_beta_share)
# truncated output for illustration
$corr_comp_vars_beta
[1] 0.5752723
$corr_macro_vars_beta
[1] 0.3024359
```

Listing 4.4: RiskAnalytics application example: QR.analytics part

Iterating through different `sel_threshold` values (c(0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.72)) via the `sapply` function, we can observe that with the increasing threshold only the macro variables remain as active regressors for the NASDAQ companies. For `sel_threshold` = 0.7 only the

macro variables 101 and 102 serve as regressors for 14 and 38 companies, respectively. Reaching `sel_threshold` = 0.72, the macro variable 101 (VIX) vanishes, what means that it is a regressor of maximally 9 companies, whereas the macro variable 102 (S&P500) is still an active regressor for some 16 companies.

While `QR.regressors.stats` provides the frequency of the covariates based on the active set of the beta coefficients, `QR.beta.stats` analyzes the beta coefficients themselves. Basically, `QR.beta.stats` calculates the average percentage (over all moving windows and companies) of active beta coefficients for the covariates. The average percentage of active beta coefficients with the value 0.2, for instance, would mean that the covariate, which has this percentage, acts as an active regressor in exactly 20% of all moving windows (n_w) averaged over all companies. The vector of the average percentage of active beta coefficients is stored in the variable `ave_beta_share` (each element corresponds to a covariate), see Listing 4.4. Besides the corresponding plots and box plots for the NASDAQ companies and macro variables, which are provided by `QR.beta.stats` based on `ave_beta_share` (see also Figure 4.5), `ave_beta_share` can be subjected to further statistical analysis. For example, `ave_beta_share` is minimal (= 0.095) for the company with the number 70 (Loews Corporation (L)) and is maximal (= 0.484) for the company with the number 88 (CBRE Group, Inc. (CBG)). The distribution of the `ave_beta_share` values of the macro variables is provided in Figure 4.5 and Table 4.3.

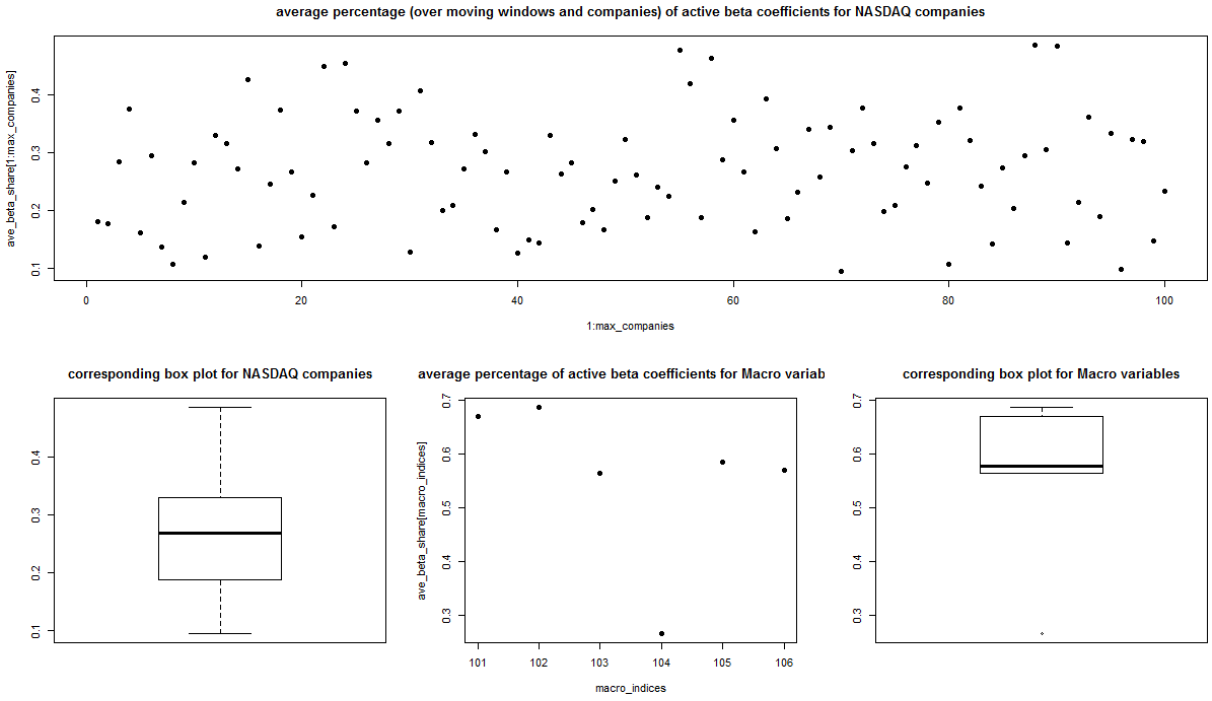


Figure 4.5: Average percentage (over moving windows and companies) of active beta coefficients for NASDAQ companies and macro variables and the corresponding box plots, produced via *QR.beta.stats*

	^VIX	^GSPC	IYR	3MTCM	Yield	Credit
Variance	0.0187	0.0042	0.0032	0.0013	0.0447	0.0227
Beta_share	0.6695	0.6865	0.5635	0.2651	0.5840	0.5698

Table 4.3: Variances versus average percentage of active beta coefficients of the macro variables, produced via *QR.variance.vs.beta*

The statistical analysis provided by the functions `QR.regressors.stats` and `QR.beta.stats` reveals that the macro variables have a dominant effect on the regressed companies. Except the macro variable with the number 104 (3MTCM: the changes in the three-month Treasury bill rate) all other macro variables have an average percentage of active beta coefficients of at least 56%. The two most influential regressors are the variables 101 and 102 (VIX and S&P500). Averaged over all moving windows and regressed companies, VIX and S&P500 are present in around two thirds of the performed quantile regressions.

An interesting observation is the relationship between the variances of the covariates and the average percentages of active beta coefficients as calculated in `ave_beta_share`. The function `QR.variance.vs.beta` examines this issue. Among other details, this function delivers the correlations between the variances and the `ave_beta_share` values (0.575 for the companies and 0.302 for macro variables, see the last part of Listing 4.4), the corresponding plots and scatter plots in Figure 4.6, and the output for Table 4.3. In particular, the scatter plot in Figure 4.6 illustrates the positive correlation between the variances of the NASDAQ companies and the corresponding average percentages of active beta coefficients, i.e. companies with higher volatility are tendentially more often active regressors with non-zero beta coefficients.

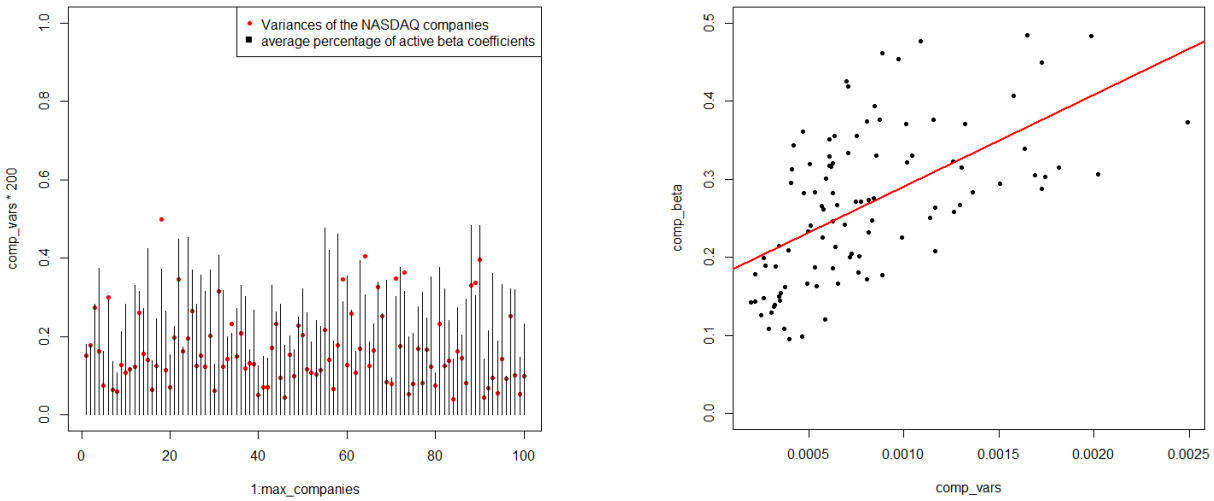


Figure 4.6: Variances versus average percentage of active beta coefficients of the NASDAQ companies: as a multiple plot with rescaled variances by factor of 200 on the left, and as a scatter plot with linear regression on the right, produced via *QR.variance.vs.beta*

4.3.4 RiskAnalytics package: "Risk Analytics" part

Listing 4.5 shows how the QR calculation results from the parallel computing component of **RiskAnalytics**, which are saved in the `parResult` object, are aggregated and the *FRM* risk measure as proposed in the "FRM methodology and estimation" section in Yu et al. (2017) is constructed. It is recalled that the *FRM* risk measure is defined as the averaged lambda over all k NASDAQ companies:

$$FRM(t) \overset{def}{=} \frac{1}{k} \sum_{j=1}^{k} \lambda_j^*(t), \quad t \in \{t_0, \ldots, T\}. \tag{4.2}$$

The function `aggregate.parallel.results` serves the purpose of combining the λ_j^*-values from each company and applying Formula 4.2. Additionally, a previous lambda time series can be read in from a CSV file and concatenated with the new lambda values counting `new_days`

entries. Finally, the current lambda time series is saved as a CSV file and returned as the vector `last_lambda` for further analysis. Subsequently, the function `lambda.analytics` provides as part of the "Risk Analytics" software component descriptive statistics for the current lambda time series `last_lambda`, furthermore λ quantiles corresponding to the risk level probabilities as suggested in the "Risk levels" section in Yu et al. (2017), the last λ with its quantile probability, and the correlations between λ and the macro variables are calculated. Finally, a simple plot preview of the FRM lambda time series is generated, see Figure 4.7.

```
#------------------------------------------------
# Aggregate data
#------------------------------------------------
last_lambda = aggregate.parallel.results(final_data, max_companies, parResult,
     work_dir = work_dir, new_days = 2469, winsize = 63)

#------------------------------------------------
# Risk Analytics
#------------------------------------------------
lambda.analytics(last_lambda, final_data, max_companies)
# "Lambda Analytics Summary"
#    Min.  1st Qu.  Median    Mean  3rd Qu.    Max.
# 0.004418 0.006351 0.007255 0.009206 0.009903 0.032530
# "Lambda quantiles corresponding to the given probabilities:"
#         0%        20%        40%        60%        80%       100%
# 0.004418067 0.006227005 0.006791834 0.008091999 0.010884704 0.032527493
# "last lambda value is the quantile for this probability: 0.384927066450567"
# "last lambda value: 0.00673865282581167"
# "Correlation between Lambda and macro variables"
#     ^VIX      ^GSPC        IYR     3MTCM    Yield   Credit
# 0.8196245 -0.01427044 -0.01283302 -0.0365718 0.2497622 0.6803068
```

Listing 4.5: RiskAnalytics application example: "Risk Analytics" part

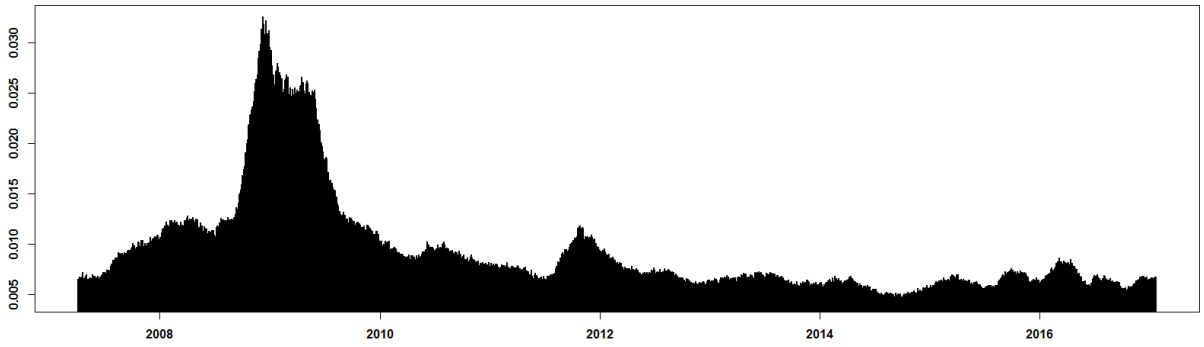


Figure 4.7: Simple plot preview of the FRM lambda time series, generated after the full program run of the **RiskAnalytics** package

4.3.5 RiskAnalytics package: full program run

The full program run of the package **RiskAnalytics** is demonstrated in Listing 4.6. The *data processing component* extracts all needed data in real time, which are passed over to the *parallel computing component*. The latter performs the lasso penalized QR (*QR methods component*) via cluster computing (**snowfall** (Knaus, 2015)), thereby operating as an "in-memory application". That means that only the computational power of the physical CPU cores is needed and no disk Input/Output operations are required. Subsequently, the parallelization results are aggregated and the *FRM* risk measure is calculated. In conclusion, the *"Risk Analytics" component*,

which comprises the tools *data.analytics*, *QR.analytics* and *lambda.analytics*, provides descriptive statistics of the data collected and calculated at different stages of the **RiskAnalytics** program run, hence helping to analyze, evaluate and forecast the systemic risk for the considered markets (Nasdaq Stock Market).

```
library(snow)
library(RiskAnalytics)

work_dir = "c:/r/frm/2017"
max_companies = 100

# Load data
companylist = get.nasdaq.companies()
system.time( yahoo_data <- get.yahoo.data(companylist, max_comp_num = max_companies, from_date =
     "2006-12-29") )
system.time( macro_data <- get.macro.data(from_date = "2006-12-28") )
final_data = combine.data(yahoo_data, macro_data)

# Calculate data
parResult = parallel.lasso.computation(final_data, max_companies, work_dir = work_dir, new_days
    = 2469, parallel_cpu = 32, winsize = 63)
# Aggregate data
last_lambda = aggregate.parallel.results(final_data, max_companies, parResult,
     work_dir = work_dir, new_days = 2469, winsize = 63)

# Risk Analytics / QR.analytics
data.analytics(yahoo_data, macro_data)
QR.regressors.stats(parResult, sel_threshold = 0.5, min_regressed_comp = 10)
ave_beta_share = QR.beta.stats(parResult)
QR.variance.vs.beta(final_data, ave_beta_share)
lambda.analytics(last_lambda, final_data, max_companies)
```

Listing 4.6: RiskAnalytics application example: full program run

4.4 RiskAnalytics (scientific IDE)

The ***RiskAnalytics*** *scientific IDE* is available under `http://borke.net/RiskAnalytics`. IDE stands for "integrated development environment". This interactive and web based IDE has the purpose of combining and presenting the scientific, technical and visual materials, elements and sources around the topic "Risk Analytics and FRM". It provides different risk meter designs both for the risk indicators and for the time series visualizations, containing current but also previous risk measure calculations. Further, scientific references concerning the methodology but also software implementations can be found within the ***RiskAnalytics*** *scientific IDE*.

Interactive exploratory data analysis (EDA) can be conducted with the aid of the D3 (Bostock et al., 2011) based risk measure visualizations, current Google Trends statistics and real-time charts (encompassing VIX, S&P 500, Nasdaq etc.), see also Figure A.9. The real-time charts are provided by TradingView, a social network for traders and investors on Stock and Futures and Forex markets (`https://www.tradingview.com/chart`).

4.5 Future Developments

4.5.1 More D3/C3 visualizations based on the beta structure

The powerful capabilities and features of the D3.js framework but also the C3.js extension, a D3-based reusable chart library (`http://c3js.org/`), can be used to implement more interactive designs and visualizations of the risk measures. For instance, the rich structure of the QR-components, lambdas and beta coefficients as time dependent vectors and matrices, can be exploited for the generation of time-variant risk dependency graphs, where the beta coefficients serve as proxies for the adjacency matrix of the systemic risk. First steps within R can be easily done by means of the package **networkD3** (Gandrud et al., 2016), see also `https://github.com/Quantlet/forceNetwork`.

4.5.2 Package namespace

The current **RiskAnalytics** package could be improved by using a namespace. Name-spaces make a package self-contained in two ways: the **imports** and the **exports** behavior. The **imports** defines how a function in one package finds a function in another. The **exports** helps to avoid conflicts with other packages by specifying which functions are available outside of the package (internal functions are available only within the own package and can't easily be used by another package). For more details, the book "R packages"[6] (Wickham, 2015) is recommended. Furthermore, a package namespace could help to reduce redundant arguments, which are passed to several functions (see e.g. `parallel.lasso.computation`, `aggregate.parallel.results`), by storing the relevant variables in a namespace, from where they can be accessed from other functions without being explicitly provided as redundant arguments.

4.5.3 Incorporation of the *hqreg* package

The aforementioned **hqreg** (Yi, 2016) package, which provides efficient and C optimized algorithms for fitting regularization paths for lasso or elastic-net penalized regression models with Huber loss, quantile loss or squared loss, is a promising alternative for the time-intensive lasso penalized QR procedure, see Section 4.3.2. A further version of the **RiskAnalytics** package could provide different lasso penalized QR implementations, with **hqreg** as a possible option. But first the necessary studies and benchmarks should be carried out in order to compare the numerical consistency, reliability and time complexity with the former methods and results.

4.5.4 More risk measures involving the beta coefficients and the market volatility

The results from Section 4.3.3 indicate that there is a considerable relationship between the variances of the covariates and the average percentages of active beta coefficients, i.e. covariates with higher volatility are tendentially more often active regressors with non-zero beta coefficients. Because the L_1-norm penalty in Formula (1.1) in Li and Zhu (2008) shrinks the fitted coefficients toward zero by $|\beta_1| + \ldots + |\beta_p| \leq s$, there is a duality between the λ value and the shrinkage parameter s of the β's L_1-norm. Hence, the incorporation of the whole market volatility (in the given moving window or another time period) and some appropriate transformations of the β-coefficients in the new risk measure variants should be considered and examined. The ***RiskAnalytics*** *scientific IDE* is a good platform for further experiments.

[6] `http://r-pkgs.had.co.nz/namespace.html`

4.6 Conclusion

The presented **RiskAnalytics** package (Borke, 2017b) is a convenient tool with the purpose of integrating lasso penalized quantile regression methods with full solution paths and cluster computing support around the topic "Risk Analytics and FRM". Its main goal is to provide data processing and parallelized quantile lasso regression methods for risk analysis based on NASDAQ data, Yahoo Finance data and some macro variables. The derived "Risk Analytics", which comprise the methods *data.analytics*, *QR.analytics* and *lambda.analytics*, can help to forecast and evaluate the systemic risk for the corresponding markets.

Supplementary R codes are published on `www.quantlet.de` with the keyword FRM. The visualization and the up-to-date FRM are available on the website `http://frm.wiwi.hu-berlin.de`. Additionally, the interactive and web based ***RiskAnalytics*** *scientific IDE*[7] combines the scientific, technical and visual materials, elements and sources around the research field "Risk Analytics". It is a good platform for further experiments and developments as discussed in Section 4.5.

> To understand computations in R, two slogans are helpful: Everything that exists is an object. Everything that happens is a function call.
>
> *John M. Chambers*

[7] `http://borke.net/RiskAnalytics/`

5 Conclusion

This study endeavoured to characterize the driving factors and parametrizations for transforming GitHub's Big Data into Smart Data. The first essential step of our work was to adjust the application program interface (API) of GitHub to the R software environment. This was accomplished by the new package **rgithubQ**. The next step was to specify a suitable language for the metadata representation. We propose the usage of YAML, a human friendly data serialization standard, as annotation language for OSR metadata, not least because there are already numerous GitHub organizations using YAML. The novel **yamldebugger** package was written in order to facilitate a standardized validation and analysis of YAML annotated OSR. The following step was to adjust and calibrate the TM models to the OSR metadata. We examined the basic vector space model (VSM) and two popular generalized VSM representations: GVSM(TT) and LSA. For that purpose, the new **TManalyzerQ** package was developed. The latter two TM models incorporate term-term correlations and semantics, thereby providing considerable sparsity reduction and achieving higher clustering performance. The main advantage of LSA is the dimension reduction property. The benchmark results showed that the LSA model seems to be applicable for Big Data and has a modest time complexity. Thus, metadata samples of about 100.000 GitHub organizations (nearly one-tenth of the GitHub universe) could be processed within this model on a single CPU in less than one hour.

Hence, the packages **rgithubQ**, **yamldebugger** and **TManalyzerQ** form the basis for a self-contained "GitHub Mining infrastructure in R" (abbreviated as *GiHuMiR*). Based on *GiHuMiR*, we introduced the **Validation Pipeline** (*Vali-PP*), a functional multi-staged instrument for clustering analysis, providing multivariate statistical analysis of the co-occurrence distribution of driving factors of the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$. This setup performs dynamic calibration of metadata configurations, TM models, clustering methods and clustering quality indices. The *Smart-Vali-PP* software framework, being a component of the overall *Vali-PP* infrastructure, allows to identify and quantify the clustering quality for a given *Vali-PP*-configuration. It can be applied to each of the 27 internal quality indices offered by the package **clusterCrit**. Using the C optimized **clusterCrit** package, having different dimensions for computation scalability (metadata, TM models, cluster sizes) and building on top of the cluster and parallel computing architecture of the **snow** package, the *Vali-PP* framework is well-equipped for the "smart clusterization" of the Big Data of OSR. Based on YAML sample data from QuantNet, the optimal *Vali-PP*-configurations for the examined quality indices could be classified into three different categories: GVSM HC, SMART GVSM and MAX HC, which demonstrated that generalized vector space models (GVSM including LSA), accurate and comprehensive metadata (SMART, MAX) and hierarchical clustering maximize the clustering quality. The experimental results also revealed that the co-occurrence frequency of the metadata configurations II (maximum) and III (smart) is high (and even reached 100% for the indices: Wemmert-Gancarski, Ball-Hall, Ratkowsky-Lance, Calinski-Harabasz, Trace-W, Xie-Beni and Dunn) among the optimal *Vali-PP*-configurations, which underlined the importance of metadata for the clustering quality.

In the following, the most important features of the introduced R packages, their interaction and some application scenarios will be outlined. The **TManalyzerQ** can be directly connected to the parser layer of *GiHuMiR* and run as an R based search engine with three implicit TM models. Due to the general approach, any set of text data can serve as an object of text analysis. The **yamldebugger** acts as a Smart Data extraction layer and can be omitted or

replaced by another text preprocessing component if another type of data is involved. Within the validation benchmark $M^4_{d_1,d_2,d_3,d_4,max}$, the packages **yamldebugger** and **TManalyzerQ** were used to produce the metadata configurations and TM models. Due to the generality of the approach, the *Vali-PP* can be applied to any set of metadata/TM model configurations of OSR data delivered by *GiHuMiR*.

The **rgithubQ** package enables a GitHub wide search for OSR using the GitHub Search API and allows to implement different parsers for data extraction from various software repositories. Performing similar as Google, it is designed to find results which are ranked by best match. Due to some lightweight parsers within the package, basic mining tasks on the QuantNet and CRAN organizations can be performed directly in the R console by use of the **rgithubQ** without further *GiHuMiR* R components.

The presented QuantNet Test Collection obtained from YAML metadata and Google Analytics was used to evaluate and calibrate the IR performance by means of the **TManalyzerQ** package. By generating three performance measurement matrices: the number of retrieved documents, the number of retrieved and relevant documents (true positives), and the precision value for each query × TM model combination, the **TManalyzerQ** constitutes a convenient tool for IR design and performance analysis. Any set of OSR metadata which can be processed by the well known R package **tm** can serve as a collection of documents for the **TManalyzerQ**. The **tm** software package was designed for applications both in research as in business intelligence tasks.

The GitHub API driven QuantNetXploRer and the corresponding D3 based visualization can be found and applied under `http://www.quantlet.de`. The driving technology behind it is Q3-D3-LSA, which is the combination of "GitHub API based QuantNet Mining infrastructure in R" (Q3), D3 implementation and LSA. The QuantNetXploRer is a working example for the functionality of the aforementioned R packages and a good demonstration for the idea of "Dynamic Clustering and Visualization of Smart Data via D3-3D-LSA" and CRR.

The Financial Risk Meter (FRM) project, which was developed by means of the QuantNet platform, stimulated the genesis of the presented R package **RiskAnalytics**. We have also shown how the functionality of this package can be extended by further C optimized implementations of algorithms computing solution paths of lasso penalized quantile regression. The identification and localization of suitable R packages in GitHub (in short: "CRAN Mining") can be easily performed by means of the **rgithubQ** package.

The D3-3D technology is a powerful combination of JavaScript libraries (D3.js and Three.js) featuring hardware acceleration for complex 2D or 3D computer animations of large data sets and enabling visual data mining and exploration in web browsers. Both of them continue to be actively developed on GitHub. D3-3D is also well suited to represent the similarity (as well as distance) of data points and the clustering geometry. The "GitHub users" and "Software Galaxies" visualizations[1] demonstrate that Three.js is capable to animate around one million data points and has the potential for Big Data exploration.

As a central result of this thesis, we have introduced a powerful "GitHub Mining infrastructure in R" (*GiHuMiR*) which allows to incorporate any GitHub organization with its content for Big Data analytics thanks to the official GitHub API. Together with the introduced **Validation Pipeline** we established an automatic OSR categorization system for data science teams.

[1] `http://borke.net/PackageNetwork/`

5.1 Future Developments

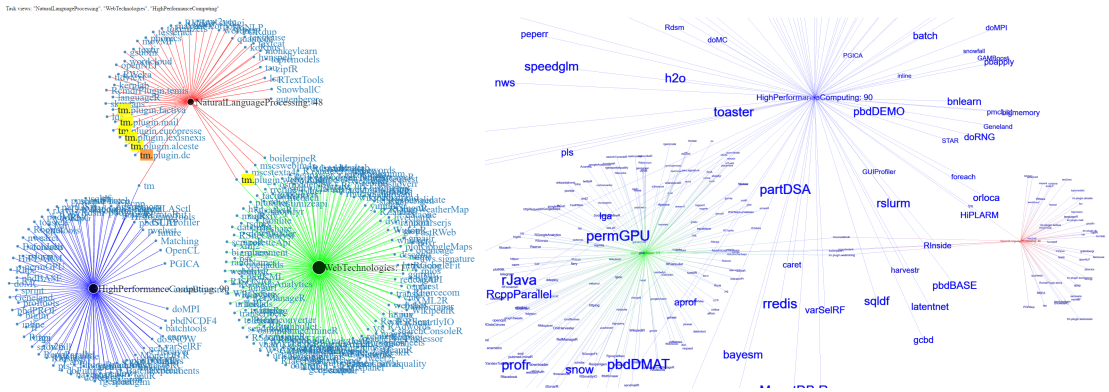


Figure 5.1: D3-3D for the CRAN Task Views "NaturalLanguageProcessing", "WebTechnologies", "HighPerformanceComputing"; the same information is visualized via D3 on the left and via 3D on the right; a click on the corresponding image opens the dynamic webpage

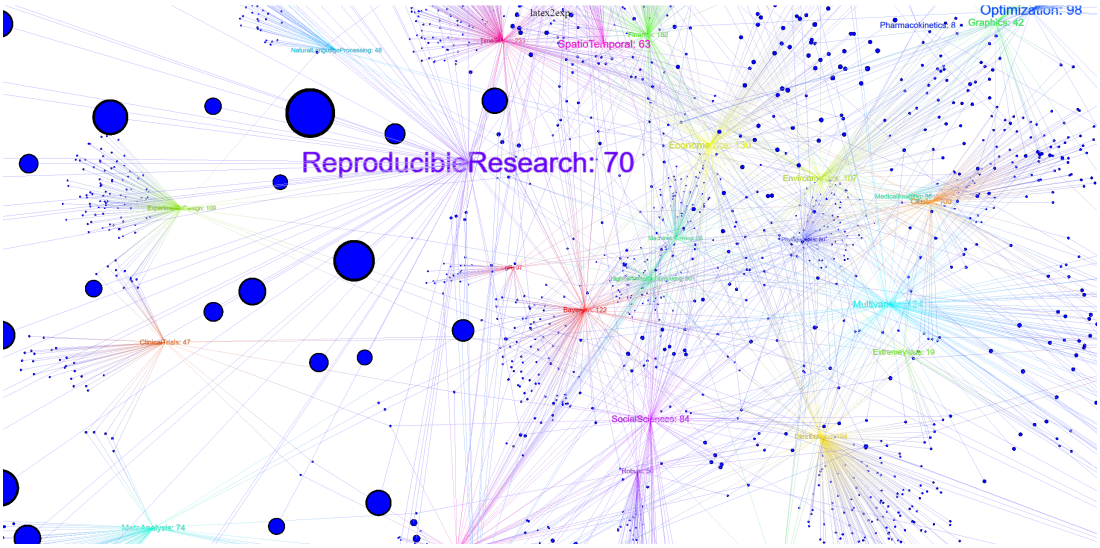


Figure 5.2: 3D for all CRAN Task Views; the Task View "Reproducible Research" containing 70 R packages is displayed in the foreground; a click on the image opens the dynamic webpage

The dynamic visualizations in Figures 5.1 and 5.2 were created by the R package **taskviewsVA** (Borke and Bykovskaya, 2017) and show various perspectives of the CRAN Task Views[2]. The **taskviewsVA** package provides Visual Analytics tools for CRAN task views and associated packages via diverse D3.js and Three.js outputs[3].

The BitQuery[4] is a GitHub API driven and D3 based search engine prototype for open source repositories. BitQuery is operated by the "GitHub Mining infrastructure in R" (and some CoffeeScript and C++ extentions) incorporating any GitHub organization and repository for Visual Analytics. Figure 5.3 provides an example showing the D3 organization in the GitHub universe.

[2] https://cran.r-project.org/web/views
[3] https://github.com/bemined/TaskviewsGenesis
[4] https://github.com/bemined/BitQuery

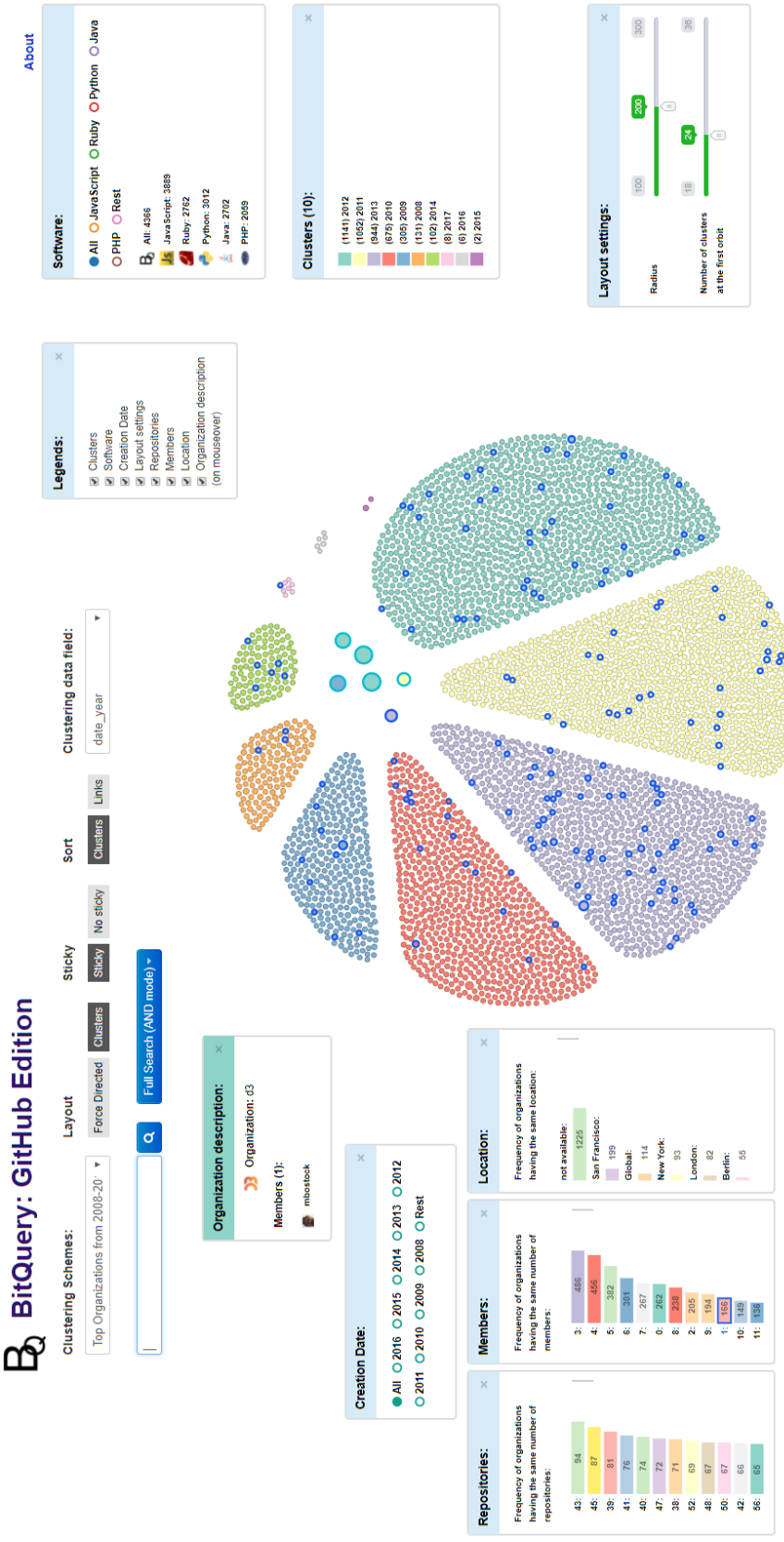


Figure 5.3: BitQuery allows interactive visual knowledge discovery of top GitHub organizations (covering the time period 2008-2013); a click on the image opens the dynamic webpage

A Appendix

A.1 Quantlet organization on GitHub

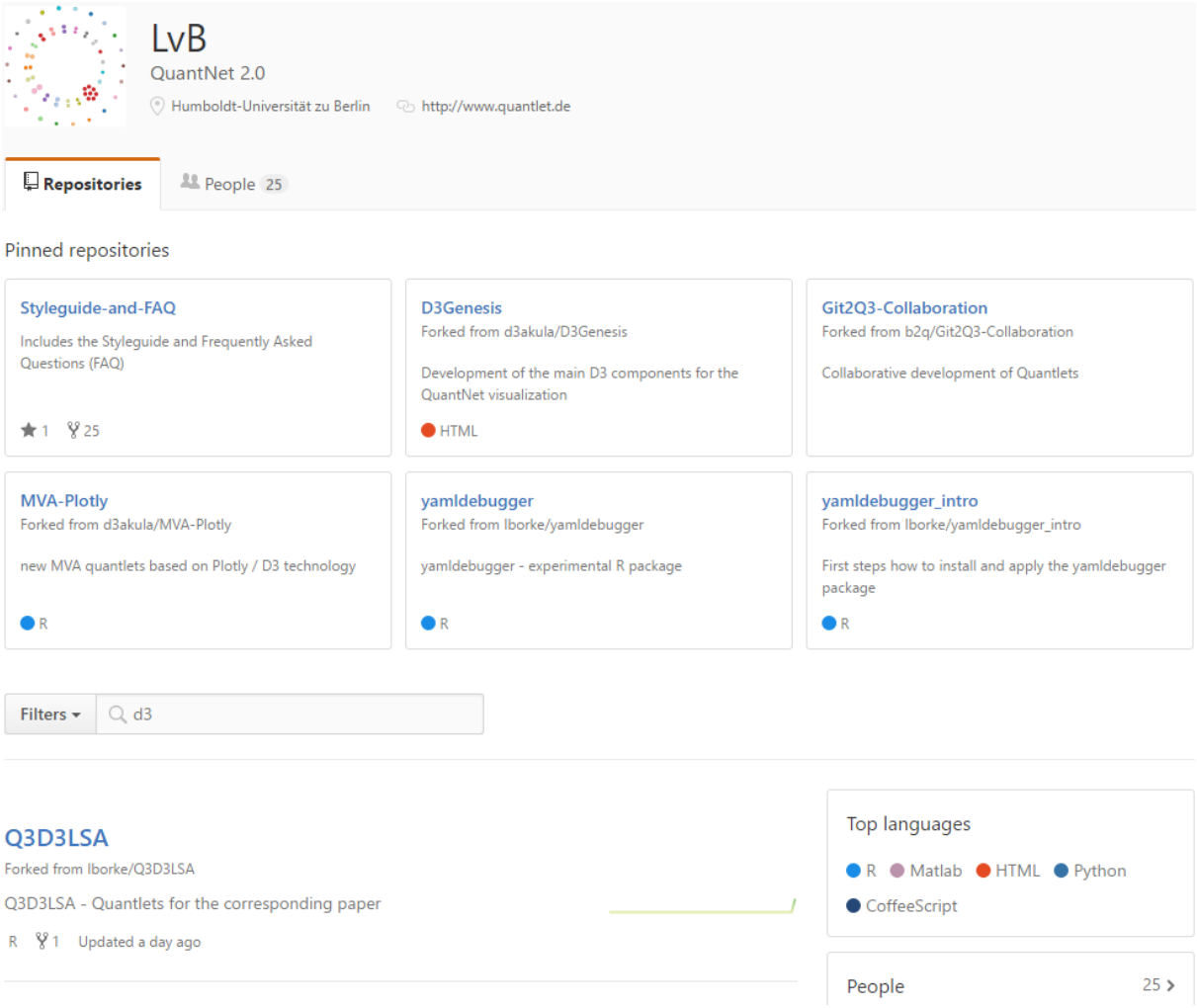


Figure A.1: Back end view: Quantlet organization on GitHub

A.2 Yamldebugger Application Example

```
> qnames = yaml.debugger.get.qnames(d_init$RootPath)
[1] "3 Q folder(s) found:"
[1] "ar1_process"  "random_walk"  "randomwalk_ar1"

> d_results = yaml.debugger.run(qnames, d_init)
[1] "1: ar1_process"
[1] "Simulates the path of a First-order autoregressive (AR-1) process over 50 ..."
[1] "Found_software: r"
[1] "Number of code files: 1 - ar1_process.R"
[1] "Number of pictures: 2 - ar1_process-1.png, ar1_process-2.png"
[1] "----------------------------------------------------------------------"
[1] "2: random_walk"
[1] "Simulates the path of a random walk over 50 time points. Epsilon terms ..."
[1] "Found_software: r"
[1] "Number of code files: 1 - random_walk.R"
[1] "Number of pictures: 2 - random_walk-1.png, random_walk-2.png"
[1] "----------------------------------------------------------------------"
[1] "3: randomwalk_ar1"
[1] "Similarity of both random walk and AR-1 (autoregressive process) to actual ..."
[1] "Found_software: r"
[1] "Number of code files: 1 - randomwalk_ar1.R"
[1] "Number of pictures: 6 - randomwalk_ar1_0.8_1.png, randomwalk_ar1_0.8_2.png, randomwalk_ar1
    _0.8_3.png, randomwalk_ar1_0.95_1.png, randomwalk_ar1_0.95_2.png, randomwalk_ar1_0.95_3.png"
[1] "----------------------------------------------------------------------"

> ( OverView = yaml.debugger.summary(qnames, d_results, summaryType = "mini") )
  Q-Quali Q folders Q Names  Descriptions stats           Keywords stats
1 A ar1_process  ar1_process 32 word(s), 157 Character(s) 9: 9 (standard), 0 (new)
2 A random_walk  random_walk 17 word(s), 93 Character(s) 9: 9 (standard), 0 (new)
3 A randomwalk_ar1 randomwalk_ar1 58 word(s), 273 Character(s) 12: 12 (standard), ...
```

Listing A.1: yamldebugger application example

A.3 Example for YAML data field analysis

```
> subset(OverView, !(`Q-Quali` %in% c("A")) )
[1] Q-Quali  Q folders  Q Names  Descriptions stats Keywords stats
<0 rows>
> as.data.frame(d_results$meta_names_distribution)
              d_results$meta_names_distribution
Author                                        3
Description                                   3
Example                                       3
Input                                         3
Keywords                                      3
Name of Quantlet                              3
Published in                                  3
See also                                      3
Submitted                                     3
> yaml.not.Qdfields(d_results$meta_names_distribution)
character(0)
> sapply( d_results$Metainfos, function(yaml){ yaml.Qdfields.nchar.from.meta(yaml) } )
   [,1] [,2] [,3]
q    11   11   14
p    20   20   20
a    11   11   11
d   222  117  370
k   137  134  177
df    0    0    0
e    82   44  417
i    85   53   85
o     0    0    0
s    31   31   32
sa   27   27   24
ce    0    0    0
cp    0    0    0
cw    0    0    0
od    0    0    0
sf    0    0    0
u     0    0    0
> rowSums(sapply( d_results$Metainfos, function(yaml){ yaml.Qdfields.nchar.from.meta(yaml) } ))
  q   p   a   d   k  df   e   i   o   s  sa  ce  cp  cw  od  sf   u
 36  60  33 709 448   0 543 223   0  94  78   0   0   0   0   0   0
> d_names = unlist(sapply( d_results$Metainfos, function(yaml){ yaml.Qdfields.from.meta(yaml)$
    found_dnames } ))
> ( d_names_distr = sort(table(d_names), decreasing = TRUE) )
d_names
 a  d  e  i  k  p  q  s sa
 3  3  3  3  3  3  3  3  3
```

Listing A.2: Example for YAML data field analysis via **yamldebugger** functions based on the results from Listing A.1

A.4 Correlation plot of YAML keywords

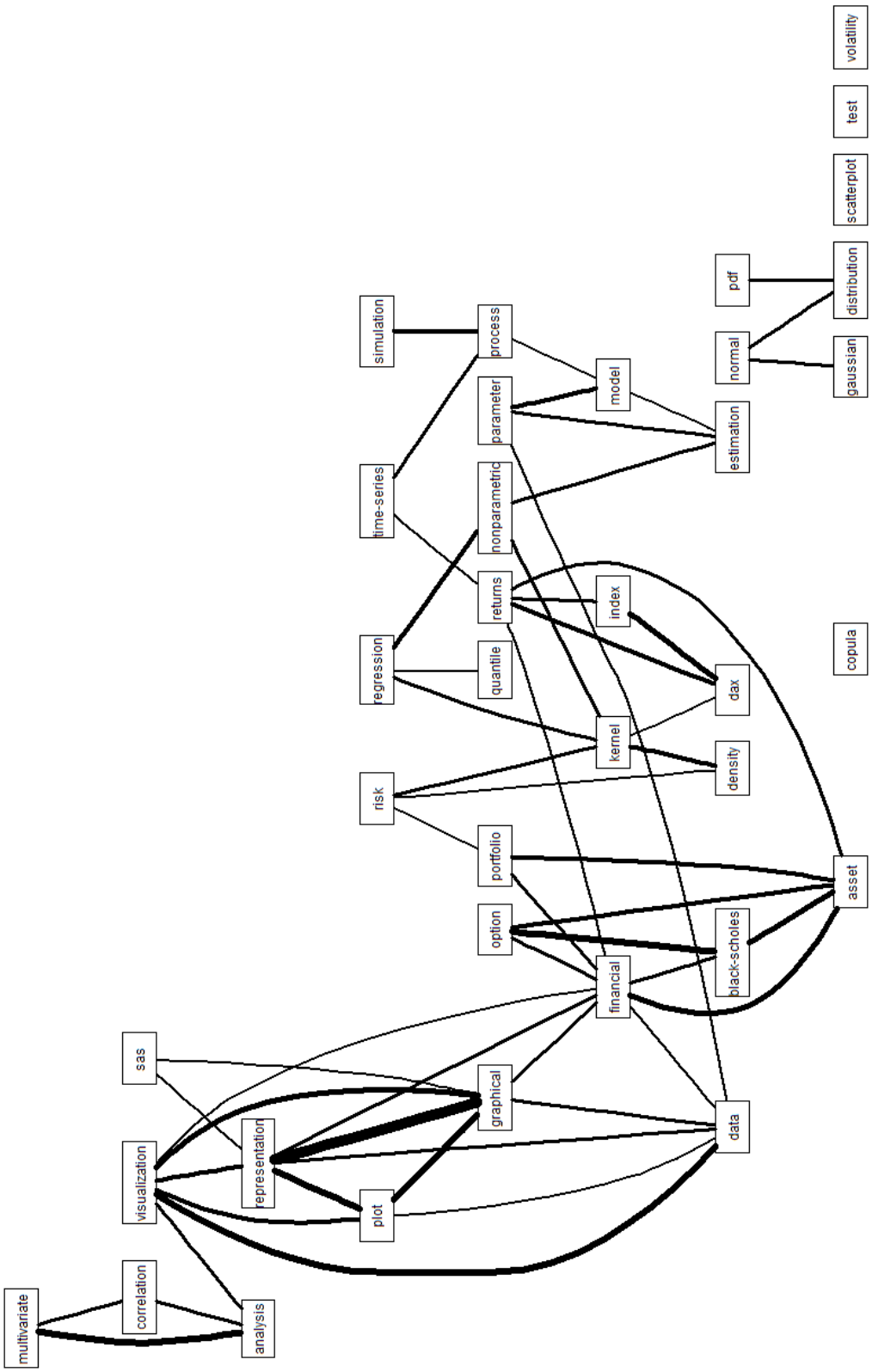


Figure A.2: Correlation plot of YAML keywords

A.5 Word clouds of YAML keywords

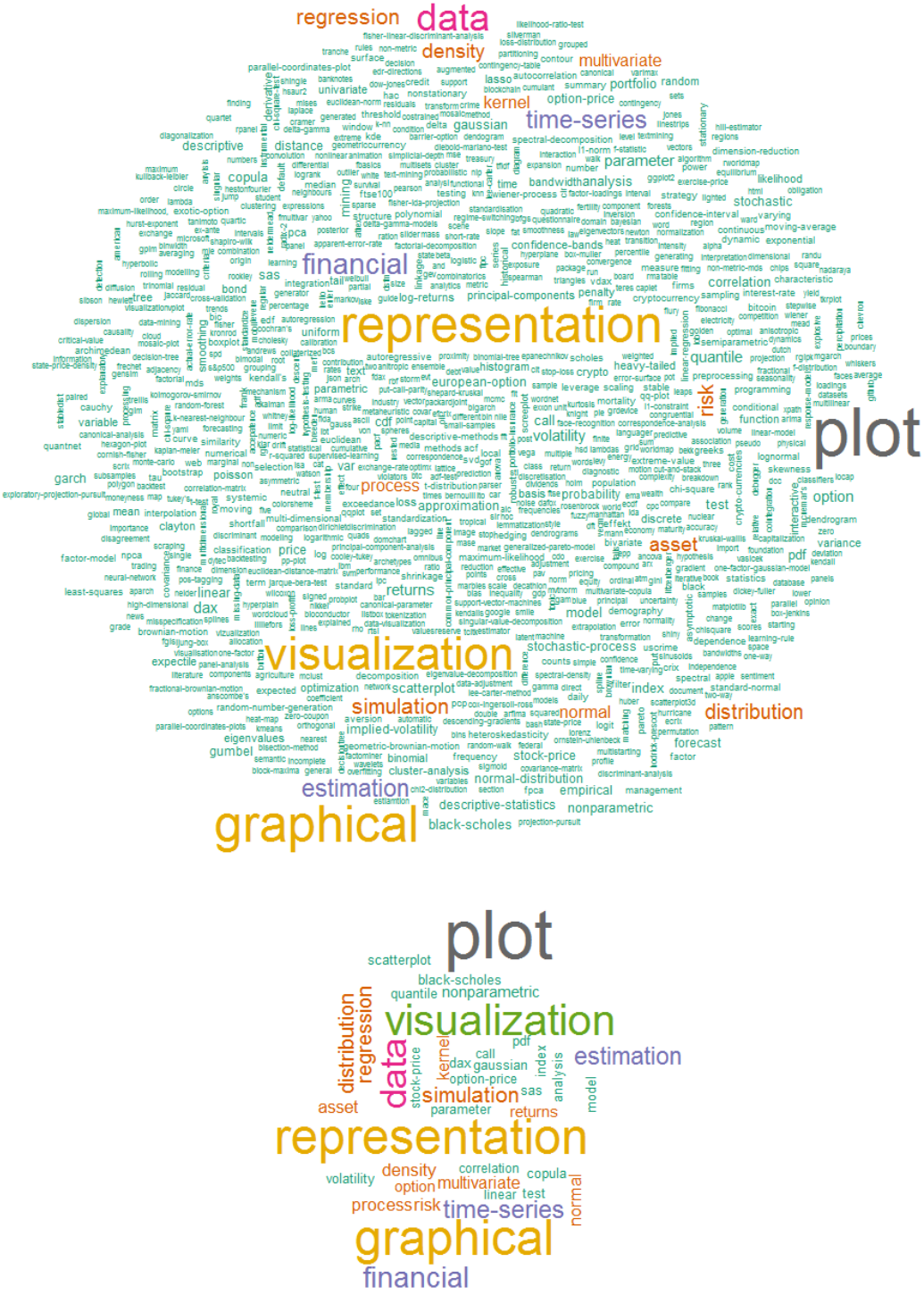


Figure A.3: Word clouds of the keywords extracted from the Quantlets' YAML meta info

The word clouds were created by yaml_wordcloud.

A.6 TManalyzerQ application example

```
library(yamldebugger)
library(TManalyzerQ)
(obj.names = load("yaml_list_full_20161122.RData", .GlobalEnv))  # 1140 Docs

t_vec = yaml.list.extract(yaml_list, weight = c(q=1, d=1, k=1, p=1))
# tf_weight = "nnc" for tf weighting; tf_weight = "ntc" for tf-idf weighting
system.time( tm_list <- tm.create.models(t_vec, tf_weight = "nnc") )
[1] "Dim TDM: 1211,1140"
[1] "Dim LSA Auto: 154"
      User     System Elapsed
     12.47       0.06      12.58

# Single term queries
query = c("covar", "random", "quantile", "histogram", "multivariate")
# Compound term queries
query = c("random number", "multivariate statistics", "black scholes")

query.tm.folded = query.tm.fold_in(query, tm_list, tf_weight = "nnc")
q_tdm_sim.tm_res = q_tdm_sim.tm.list(query.tm.folded)

# 3 threshold levels for IR
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.8)
(m1 = q_ir_list$retrieved_m)
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.7)
(m2 = q_ir_list$retrieved_m)
q_ir_list = query.similar.doc.inspect(q_tdm_sim.tm_res, sim_threshold = 0.6)
(m3 = q_ir_list$retrieved_m)
colnames(m1) = colnames(m2) = colnames(m3) = c("B", "TT", "LSA", "L50")
( m_full = cbind(m1, m2, m3) )
              B TT LSA L50 B TT LSA L50 B TT LSA L50
covar         0  0   0   7 0  0   3   9 0  0   6   9
random        0  0   0   6 0  1   0   7 0 10   3  18
quantile      0  0   0   0 0  1   0   1 0  1   2   6
histogram     0  0   0   3 0  0   2   6 2  2   4  14
multivariate  0  0   0   0 0  0   0   4 0  0   0  16

query.tm.folded$q_tdm
          Docs
  Terms    q1 q2 q3 q4 q5
  covar     1  0  0  0  0
  histogram 0  0  0  1  0
  multivari 0  0  0  0  1
  quantil   0  0  1  0  0
  random    0  1  0  0  0
```

Listing A.3: IR system designs via **TManalyzerQ**

A.7 Smart-Vali-PP multi.which

```
# A which for multidimensional arrays. Mark van der Loo 16.09.2011
# A Array of booleans
# returns a sum(A) x length(dim(A)) array of multi-indices where A == TRUE
multi.which <- function(A){
    if ( is.vector(A) ) return(which(A))
    d <- dim(A)
    T <- which(A) - 1
    nd <- length(d)
    t( sapply(T, function(t){
        I <- integer(nd)
        I[1] <- t %% d[1]
        sapply(2:nd, function(j){
            I[j] <<- (t %/% prod(d[1:(j-1)])) %% d[j]
        })
        I
    }) + 1 )
}
```

Listing A.4: *multi.which* for multidimensional arrays

A.8 CRAN Mining

	repo.path	date	version	package.author
1	CY-dev/hqreg	2017-2-15	1.4	Congrui Yi
2	cran/hqreg	2017-2-15	1.4	Congrui Yi
3	cran/cqrReg	2015-04-07	1.2	Jueyu Gao & Linglong Kong
4	bssherwood/rqpen	2017-2-01	2.0	Ben Sherwood [aut, cre], Adam M...
5	cran/rqPen	2016-11-03	1.5.1	Ben Sherwood [aut, cre], Adam M...

Table A.1: All R packages on GitHub dealing with "quantile lasso regression" with additional details like submission date, version, authors; extracted via **rgithubQ**

A.9 Smart-Vali-PP application example

```
library(clusterCrit)
source("smart_vali_pp.R")
(load("results/obj_pp_d_clusterCrit_1140Q_20161222.RData", .GlobalEnv))

(ind_names = getCriteriaNames(TRUE)[c(1,3,4,5,7,28,31,32,37,39,41,42)])
# [1] "Ball_Hall"       "C_index"          "Calinski_Harabasz" "Davies_Bouldin"
# [5] "Dunn"            "McClain_Rao"      "Ray_Turi"          "Ratkowsky_Lance"
# [9] "Silhouette"      "Trace_W"          "Wemmert_Gancarski" "Xie_Beni"

# only max/min interpretation
ind_goal_func = c(min, min, max, min, max, min, min, max, max, min, max, min)

i = 2 ; yrange = c(0, 0.4) # clusterCrit//C_index
i = 7 ; yrange = c(0, 12)  # clusterCrit//Ray_Turi
i = 11; yrange = c(0, 0.5) # clusterCrit//Wemmert_Gancarski

# 4-dim array for the selected index i : Meta conf / Methods / Models / nclusters
sel_index_vali = pp_d[i,,,,]
index.optimal.func = apply(sel_index_vali,
                           length(dim(sel_index_vali)), ind_goal_func[[i]])
optimal_share_index = optimal_share.for.index(sel_index_vali, index.optimal.func)

# Main Evaluation 1
optimal_share.prettify(optimal_share_index)
# Main Evaluation 2
# plot only graphs which intersect the best function at least at one point
plot.optimal.functions(sel_index_vali, index.optimal.func, optimal_share_index,
                       yrange, ind_names[i])
# plot ALL graphs of Vali-PP configurations
plot.optimal.functions(sel_index_vali, index.optimal.func, optimal_share_index,
                       yrange, ind_names[i], only_partly_best = FALSE)
```

Listing A.5: Smart-Vali-PP application example

A.10 Smart-Vali-PP Results

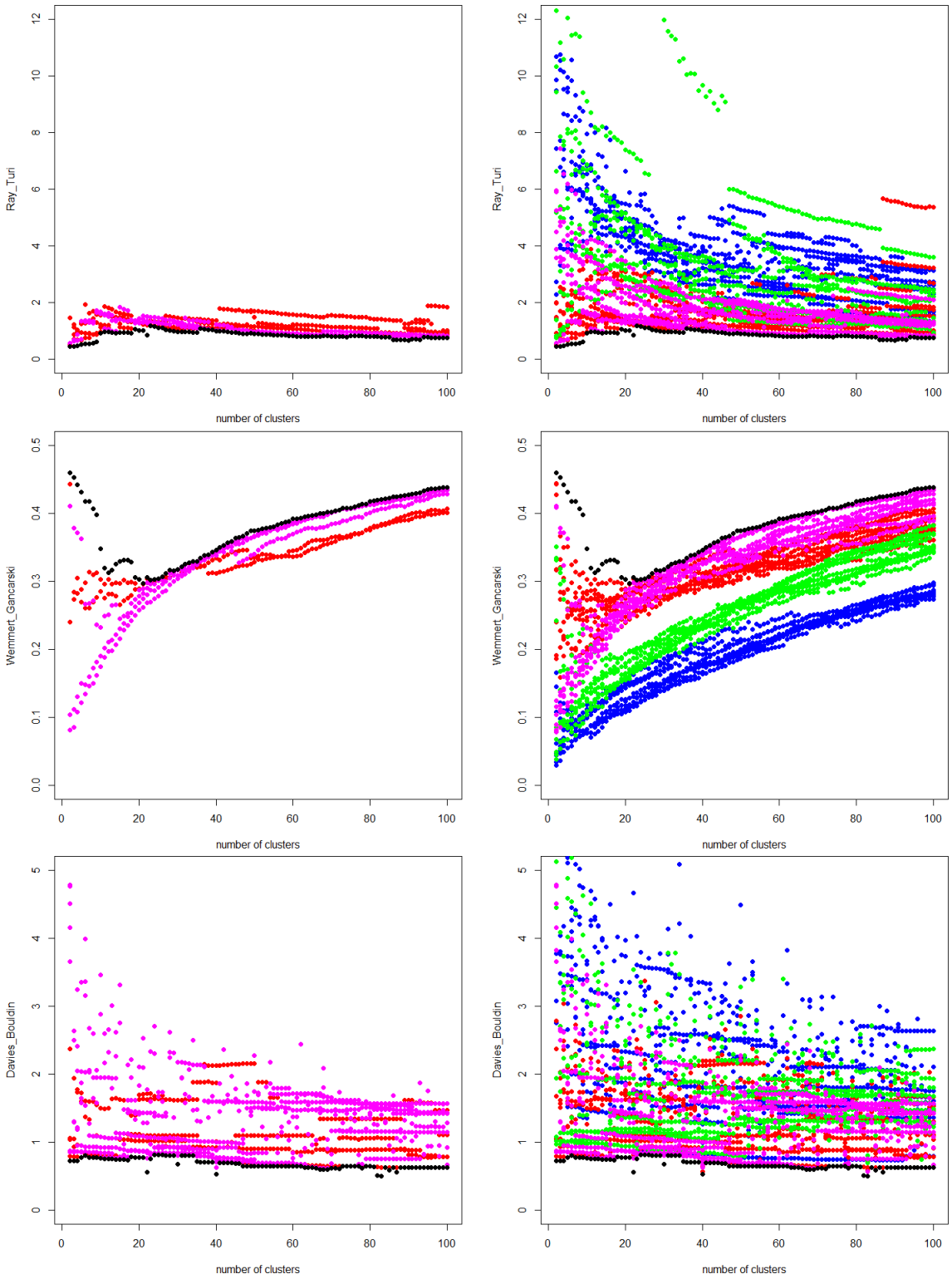


Figure A.4: *Smart-Vali-plots* of YAML-1140 for the indices: Ray-Turi, Wemmert-Gancarski, Davies-Bouldin from the **clusterCrit** package; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

Conf. d_1	Method d_3	Model d_2	Freq.	Cum. rel. prop.	Rel. prop.
Ray-Turi					
3	3	4	38	0.38	0.38
1	3	4	20	0.59	0.20
3	3	2	18	0.77	0.18
2	3	4	17	0.94	0.17
2	3	2	4	0.98	0.04
3	2	2	1	0.99	0.01
1	3	2	1	1.00	0.01
Wemmert-Gancarski					
2	3	4	68	0.69	0.69
3	3	4	20	0.89	0.20
3	2	2	5	0.94	0.05
3	3	2	3	0.97	0.03
3	2	4	2	0.99	0.02
2	2	4	1	1.00	0.01
Davies-Bouldin					
1	3	2	29	0.29	0.29
1	3	4	27	0.57	0.27
3	2	4	10	0.67	0.10
3	2	2	8	0.75	0.08
2	1	4	5	0.80	0.05
2	2	4	5	0.85	0.05
3	3	4	4	0.89	0.04
3	1	4	3	0.92	0.03
1	2	4	3	0.95	0.03
2	3	4	3	0.98	0.03
2	3	2	1	0.99	0.01
3	3	2	1	1.00	0.01

Table A.2: *Smart-Vali-tables* of YAML-1140 for the indices: Ray-Turi, Wemmert-Gancarski, Davies-Bouldin from the **clusterCrit** package

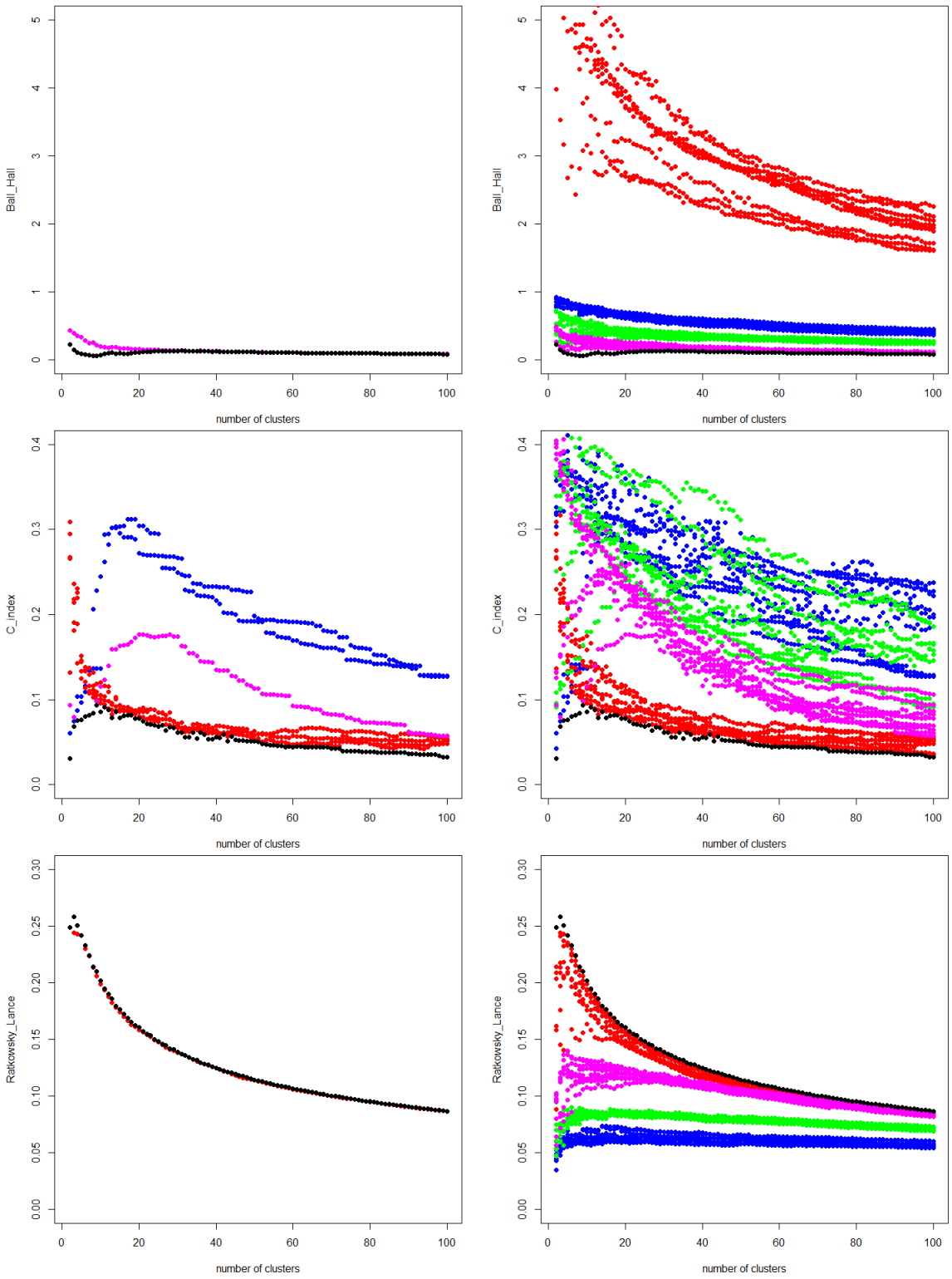


Figure A.5: *Smart-Vali-plots* of YAML-1140 for the indices: Ball-Hall, C-index, Ratkowsky-Lance from the **clusterCrit** package; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

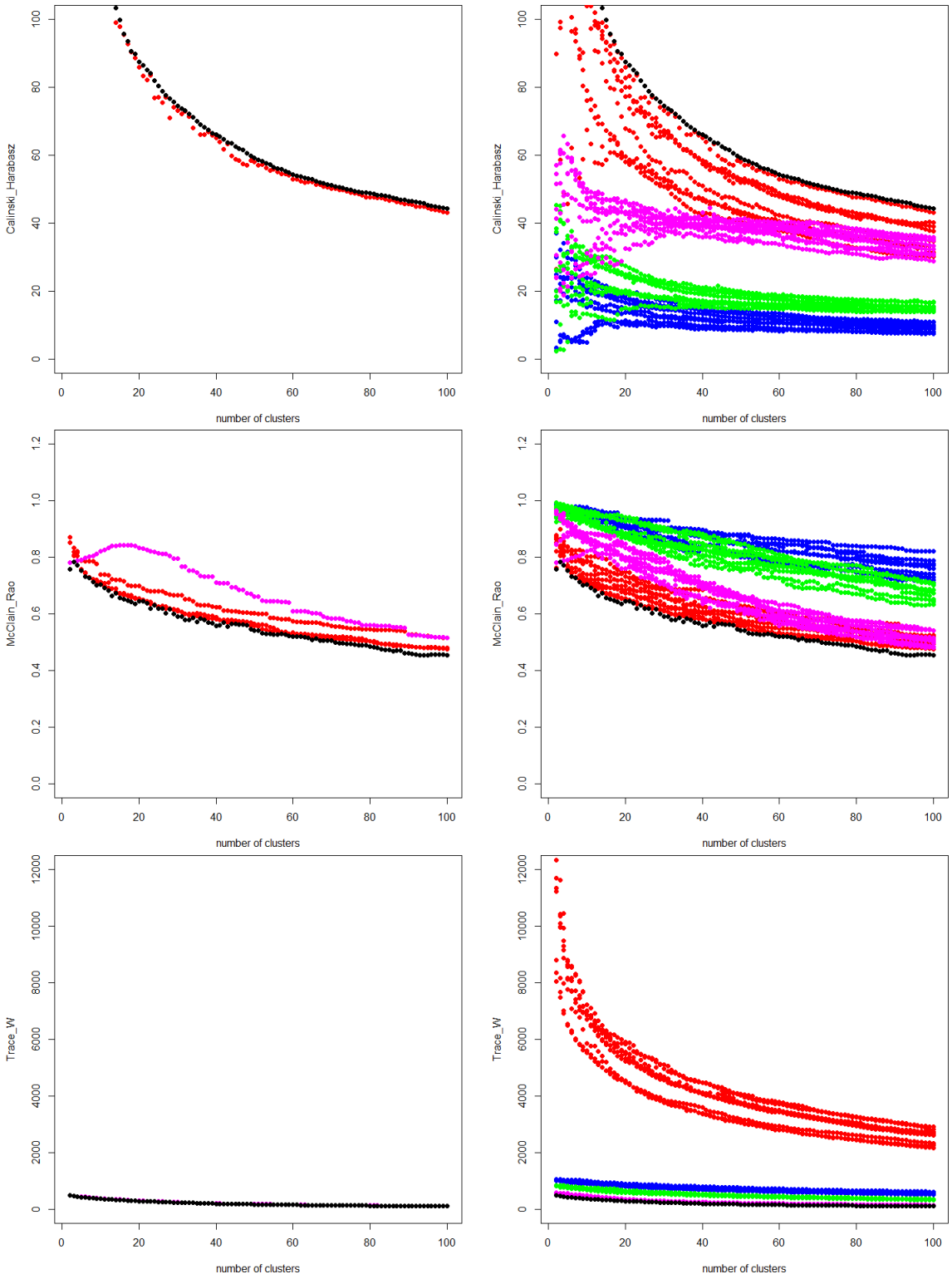


Figure A.6: *Smart-Vali-plots* of YAML-1140 for the indices: Calinski-Harabasz, McClain-Rao, Trace-W from the **clusterCrit** package; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

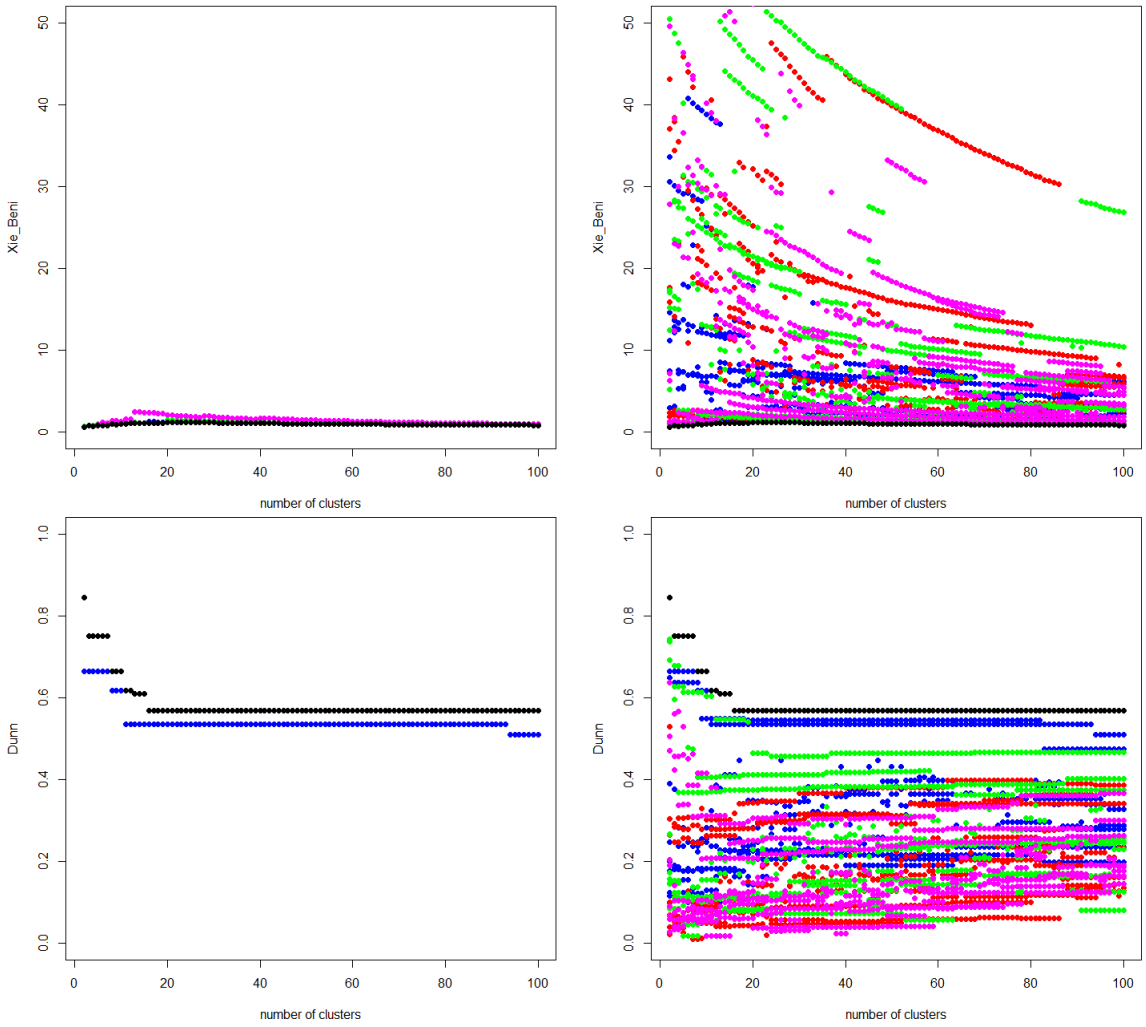


Figure A.7: *Smart-Vali-plots* of YAML-1140 for the indices: Xie-Beni, Dunn from the **clusterCrit** package; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

Conf. d_1	Method d_3	Model d_2	Freq.	Cum. rel. prop.	Rel. prop.
Ball-Hall					
2	3	4	57	0.58	0.58
2	2	4	42	1.00	0.42
C-index					
3	3	2	50	0.50	0.50
3	2	2	15	0.66	0.15
3	1	2	14	0.80	0.14
1	1	2	10	0.90	0.10
2	3	4	5	0.95	0.05
1	2	2	3	0.98	0.03
2	3	1	1	0.99	0.01
3	3	1	1	1.00	0.01
Ratkowsky-Lance					
3	2	2	64	0.65	0.65
3	1	2	35	1.00	0.35

Table A.3: *Smart-Vali-tables* of YAML-1140 for the indices: Ball-Hall, C-index, Ratkowsky-Lance from the **clusterCrit** package

Conf. d_1	Method d_3	Model d_2	Freq.	Cum. rel. prop.	Rel. prop.
Calinski-Harabasz					
3	2	2	91	0.92	0.92
3	1	2	8	1.00	0.08
McClain-Rao					
3	1	2	90	0.91	0.91
3	2	2	5	0.96	0.05
1	1	2	2	0.98	0.02
1	3	2	1	0.99	0.01
2	3	4	1	1.00	0.01
Trace-W					
2	2	4	85	0.86	0.86
2	1	4	14	1.00	0.14

Table A.4: *Smart-Vali-tables* of YAML-1140 for the indices: Calinski-Harabasz, McClain-Rao, Trace-W from the **clusterCrit** package

Conf. d_1	Method d_3	Model d_2	Freq.	Cum. rel. prop.	Rel. prop.
Xie-Beni					
2	3	1	88	0.89	0.89
2	3	3	8	0.97	0.08
2	3	4	3	1.00	0.03
Dunn					
2	3	1	96	0.97	0.97
3	3	1	3	1.00	0.03

Table A.5: *Smart-Vali-tables* of YAML-1140 for the indices: Xie-Beni, Dunn from the **clusterCrit** package

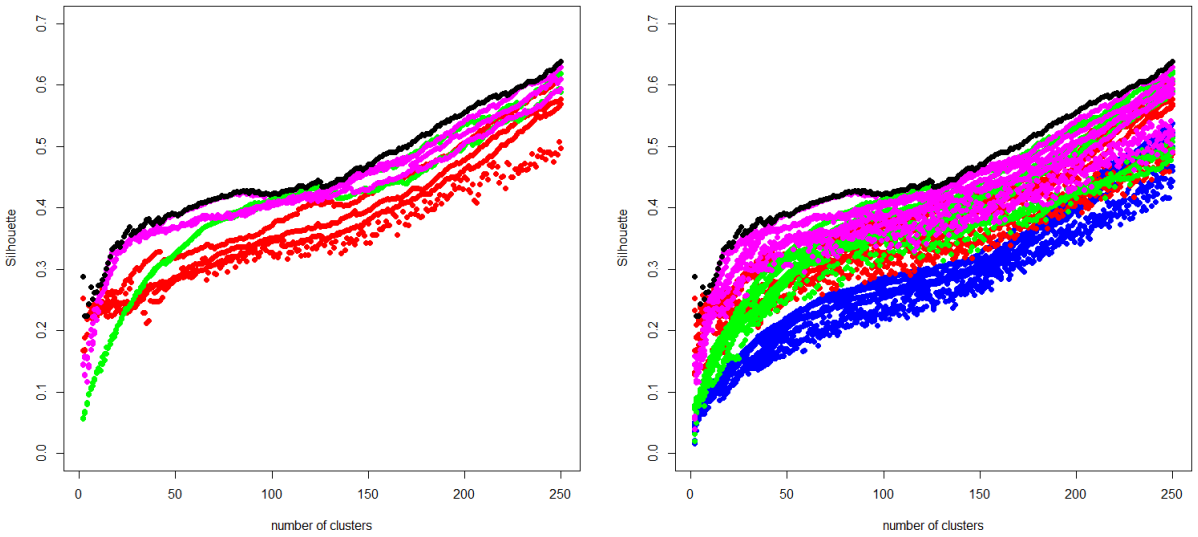


Figure A.8: *Smart-Vali-plot* of YAML-500 for the Silhouette index from the **NbClust** package; Standard TM colors: BVSM, GVSM(TT), LSA, LSA25

Conf. d_1	Method d_3	Model d_2	Freq.	Cum. rel. prop.	Rel. prop.
Silhouette					
3	3	4	119	0.47	0.47
2	3	4	82	0.79	0.32
2	2	4	28	0.90	0.11
3	2	4	11	0.94	0.04
3	3	2	7	0.97	0.03
2	3	3	3	0.98	0.01
3	1	2	1	0.99	0.00
1	2	2	1	0.99	0.00
1	3	2	1	1.00	0.00
2	2	3	1	1.00	0.00

Table A.6: *Smart-Vali-table* of YAML-500 for the Silhouette index from the **NbClust** package

A.11 *RiskAnalytics scientific IDE*

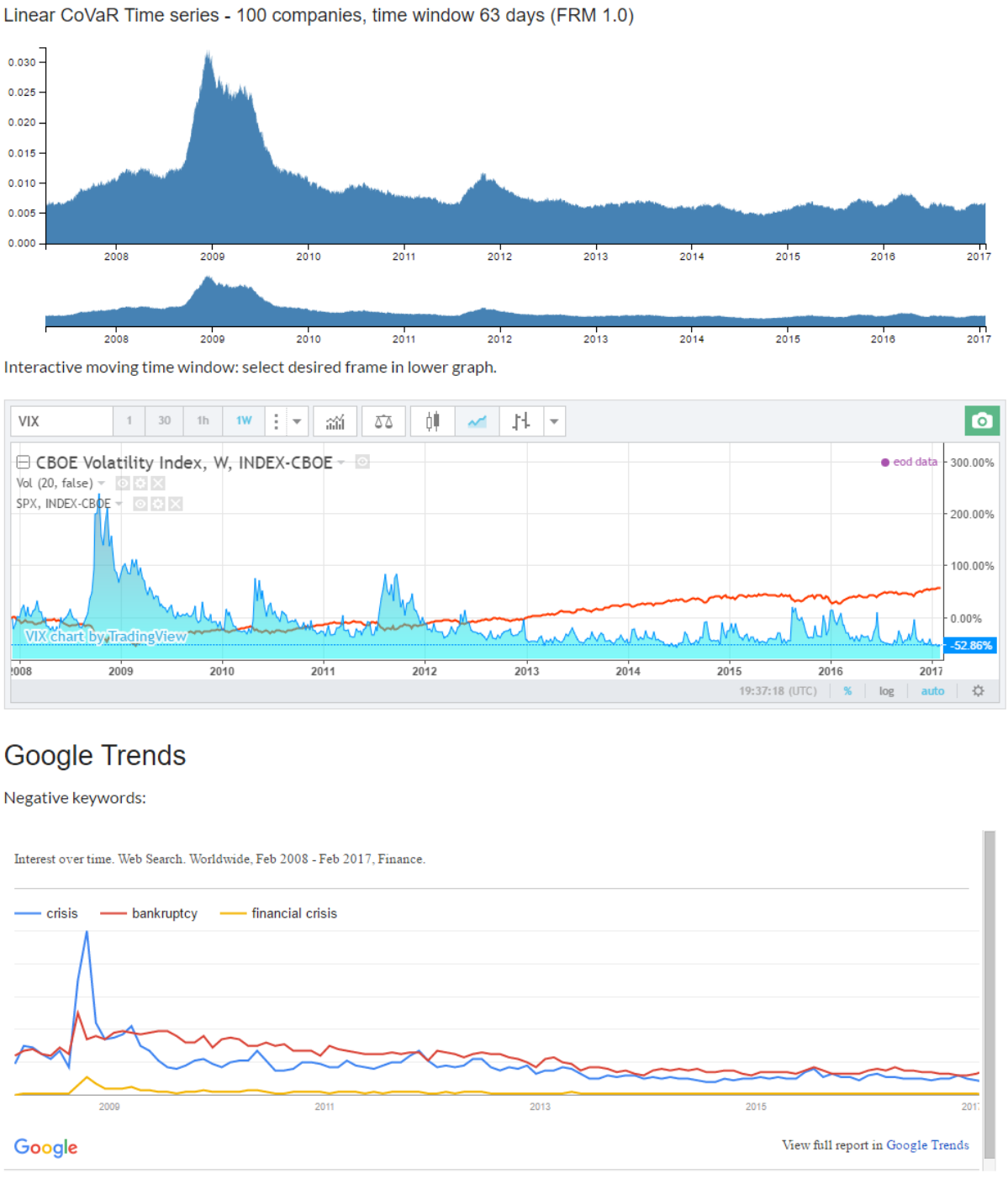


Figure A.9: D3 based *FRM* risk measure visualization (created via the **RiskAnalytics** package), real-time charts (encompassing VIX and S&P 500) and current Google Trends statistics, each of them covering the same time range; available for interactive exploratory data analysis on the ***RiskAnalytics*** *scientific IDE*

A.12 3D GitHub Network Graph

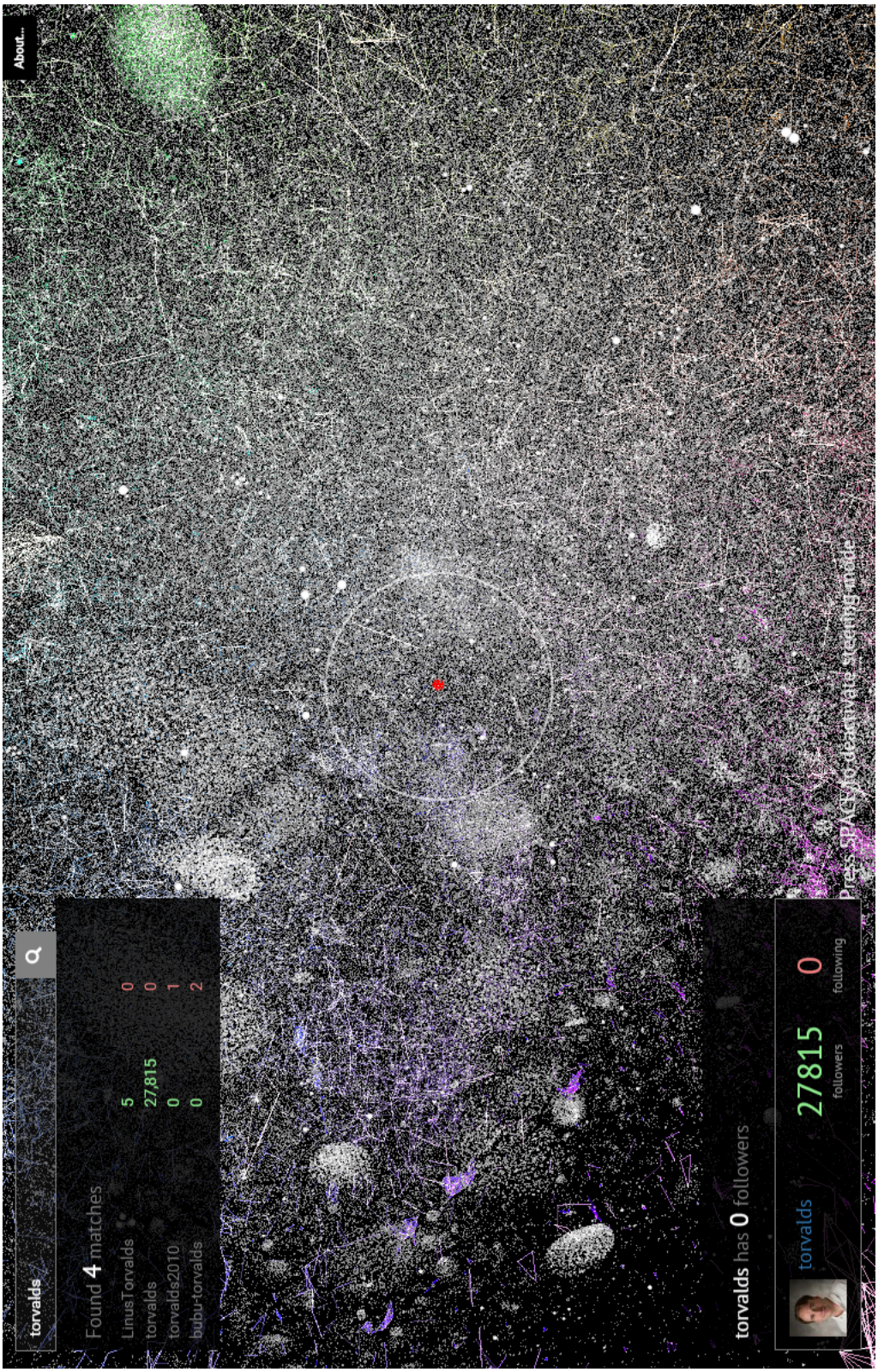


Figure A.10: 3D GitHub Network Graph: Linus Torvalds as selected user

Bibliography

Berry, M. (2003). *Survey of Text Mining: Clustering, Classification, and Retrieval*. Springer New York, 1st edition.

Borak, S., Härdle, W., and López-Cabrera, B. (2013). *Statistics of Financial Markets: Exercises and Solutions*. Springer Berlin Heidelberg, 2nd edition.

Borke, L. (2017a). RiskAnalytics: an R package for real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods. *SFB 649 Discussion Paper*. Humboldt Universität zu Berlin.

Borke, L. (2017b). *RiskAnalytics: Real time processing of Nasdaq and Yahoo finance data and parallelized quantile lasso regression methods*. R package version 0.2.0.

Borke, L. (2017c). *TManalyzerQ: Provides IR tools in 3 text mining models: BVSM, GVSM(TT) and LSA - QuantNet edition*. R package version 0.5.0.

Borke, L. (2017d). *yamldebugger: YAML parser debugger according to the QuantNet style guide*. R package version 1.0.

Borke, L. and Bykovskaya, S. (2017). *taskviewsVA: Visual analytics for CRAN task views*. R package version 0.4.0.

Borke, L. and Härdle, W. K. (2016). Q3-D3-LSA. *SFB 649 Discussion Paper*. Humboldt Universität zu Berlin.

Borke, L. and Härdle, W. K. (2017). GitHub API based QuantNet Mining infrastructure in R. *SFB 649 Discussion Paper*. Humboldt Universität zu Berlin.

Bostock, M., Ogievetsky, V., and Heer, J. (2011). D3 Data-Driven Documents. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):2301–2309.

Bradford, R. B. (2009). Comparability of LSI and Human Judgment in Text Analysis Tasks. In *Proceedings of the 11th WSEAS International Conference on Mathematical Methods and Computational Techniques in Electrical Engineering*, MMACTEE'09, pages 359–366, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

Brock, G., Pihur, V., Datta, S., and Datta, S. (2008). clValid: An R Package for Cluster Validation. *Journal of Statistical Software*, 25(1):1–22.

Charrad, M., Ghazzali, N., Boiteau, V., and Niknafs, A. (2014). NbClust: An R package for determining the relevant number of clusters in a data set. *Journal of Statistical Software*, 61(6):1–36.

Cleverdon, C. W. (1959). The evaluation of systems used in information retrieval (1958: Washington). In *Proceedings of the International Conference on Scientific Information - Two Volumes*, pages 687–698, Washington: National Academy of Sciences. National Research Council.

Cleverdon, C. W., Mills, J., and Keen, M. (1966). Factors determining the performance of indexing systems. In *ASLIB Cranfield project, Cranfield*, pages 37–59. ASLIB.

Cosentino, V., Luis, J., and Cabot, J. (2016). Findings from GitHub: Methods, Datasets and Limitations. In *Proceedings of the 13th International Conference on Mining Software Repositories*, MSR '16, pages 137–141, New York, NY, USA. ACM.

Cristianini, N., Shawe-Taylor, J., and Lodhi, H. (2002). Latent Semantic Kernels. *Journal of Intelligent Information Systems*, 18(2):127–152.

De Mauro, A., Greco, M., and Grimaldi, M. (2015). What is big data? a consensual definition and a review of key research topics. In *4th International Conference on Integrated Information, Madrid.*, pages 97–104. AIP Publishing LLC.

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by Latent Semantic Analysis. *Journal of the American Society for Information Science*, 41(6):391–407.

Desgraupes, B. (2013). *Clustering Indices*. University Paris Ouest, Lab Modal'X.

Desgraupes, B. (2016). *clusterCrit: Clustering Indices*. R package version 1.2.7.

Dunn, J. C. (1974). Well separated clusters and fuzzy partitions. *Journal on Cybernetics*, 4:95–104.

Everitt, B. S., Landau, S., Leese, M., and Stahl, D. (2011). *Cluster Analysis*. John Wiley & Sons, Ltd, 5th edition.

Feinerer, I. and Hornik, K. (2015). *tm: Text Mining Package*. R package version 0.6-2.

Feinerer, I., Hornik, K., and Meyer, D. (2008). Text Mining Infrastructure in R. *Journal of Statistical Software*, 25(5):1–54.

Feinerer, I. and Wild, F. (2007). Automated Coding of Qualitative Interviews with Latent Semantic Analysis. In Mayr and Karagiannis, editors, *Information Systems Technology and its Applications, 6th International Conference ISTA*, pages 66–77, Bonn, Germany. Gesellschaft für Informatik.

Fernández-Luna, J. M., Huete, J. F., and Rodríguez-Cano, J. C. (2011). User Intent Transition for Explicit Collaborative Search Through Groups Recommendation. In *Proceedings of the 3rd International Workshop on Collaborative Information Retrieval*, CIR '11, pages 23–28, New York, NY, USA. ACM.

Franke, J., Härdle, W., and Hafner, C. (2015). *Statistics of Financial Markets: An Introduction*. Springer Berlin Heidelberg, 4th edition.

Gandrud, C., Allaire, J., and Russell, K. (2016). *networkD3: D3 JavaScript Network Graphs from R*. R package version 0.2.13.

Golyandina, N. and Korobeynikov, A. (2014). Basic Singular Spectrum Analysis and Forecasting with R. *Computational Statistics and Data Analysis*, 71:934–954. R package version 0.14.

Gousios, G. and Spinellis, D. (2012). Ghtorrent: Github's data from a firehose. In *Mining Software Repositories (MSR), 2012 9th IEEE Working Conference on*, pages 12–21.

Handl, J., Knowles, J., and Kell, D. B. (2005). Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–12.

Härdle, W., Hautsch, N., and Overbeck, L. (2008). *Applied Quantitative Finance*. Springer Berlin Heidelberg, 2nd edition.

Härdle, W. and Simar, L. (2015). *Applied Multivariate Statistical Analysis*. Springer Berlin Heidelberg, 4th edition.

Haslwanter, T. (2016). *An Introduction to Statistics with Python: With Applications in the Life Sciences*. Springer International Publishing, 1st edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning: data mining, inference and prediction*. Springer, New York, 2 edition.

Herbert, K. G., Wang, J. T., and Liu, J. (2004). Information Retrieval and Data Mining. In Tucker, A. B., editor, *Computer Science Handbook, Second Edition*, pages 75.1–75.5. Chapman & Hall/CRC, 2nd edition.

Hilbert, M. (2016). Big data for development: A review of promises and challenges. *Development Policy Review*, 34(1):135–174.

Jones, K. S. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21.

Kalliamvakou, E., Gousios, G., Blincoe, K., Singer, L., German, D. M., and Damian, D. (2014). The Promises and Perils of Mining GitHub. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, MSR 2014, pages 92–101, New York, NY, USA. ACM.

Kaufman, L. and Rousseeuw, P. J. (2008). *Finding Groups in Data*, chapter Partitioning Around Medoids (Program PAM), pages 68–125. John Wiley & Sons, Inc.

Kaushik, A. (2010). *Web Analytics 2.0: The Art of Online Accountability and Science of Customer Centricity*. Serious skills. Wiley.

Keim, D. A. (2002). Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8.

Knaus, J. (2015). *snowfall: Easier cluster computing (based on snow)*. R package version 1.84-6.1.

Koenker, R. (2016). *quantreg: Quantile Regression*. R package version 5.29.

Koenker, R. and Mizera, I. (2014). Convex Optimization in R. *Journal of Statistical Software*, 60(1):1–23.

Korobeynikov, A. (2010). Computation- and space-efficient implementation of SSA. *Statistics and Its Interface*, 3(3):357–368. R package version 0.14.

Korobeynikov, A., Larsen, R. M., and Laboratory, L. B. N. (2016). *svd: Interfaces to Various State-of-Art SVD and Eigensolvers*. R package version 0.4.

Lang, D. T. and the CRAN team (2016). *RCurl: General Network (HTTP/FTP/...) Client Interface for R*. R package version 1.95-4.8.

Lesk, M. and Salton, G. (1968). Relevance assessments and retrieval system evaluation. *Information Storage and Retrieval*, 4(4):343 – 359.

Li, Y. and Zhu, J. (2008). L1-Norm Quantile Regression. *Journal of Computational and Graphical Statistics*, 17(1).

Linstead, E., Rigor, P., Bajracharya, S., Lopes, C., and Baldi, P. F. (2008). Mining Internet-Scale Software Repositories. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems 20*, pages 929–936. Curran Associates, Inc.

Loeliger, J. (2009). *Version Control with Git - Powerful Tools and Techniques for Collaborative Software Development*. O'Reilly Media, Inc., Sebastopol.

Maechler, M., Rousseeuw, P., Struyf, A., Hubert, M., and Hornik, K. (2016). *cluster: Cluster Analysis Basics and Extensions*. R package version 2.0.5.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA.

Michailidis, G. (2008). Data Visualization Through Their Graph Representations. In *Handbook of Data Visualization*, Springer Handbooks of Computational Statistics, chapter 5, pages 103–120. Springer Berlin Heidelberg, Berlin, Heidelberg.

Miller, T., Klein, B., and Wolf, E. (2009). Exploiting Latent Semantic Relations in Highly Linked Hypertext for Information Retrieval in Wikis. In *Proceedings of the International Conference RANLP-2009*, pages 241–245. Association for Computational Linguistics.

Mohamed, M. and Oussalah, M. (2014). *Proceedings of the First AHA!-Workshop on Information Discovery in Text*, chapter A Comparative Study of Conversion Aided Methods for WordNet Sentence Textual Similarity, pages 37–42. Association for Computational Linguistics and Dublin City University.

North, S., Scheidegger, C., Urbanek, S., and Woodhull, G. (2015). Collaborative visual analysis with RCloud. In *Visual Analytics Science and Technology (VAST), 2015 IEEE Conference on*, pages 25–32.

Pearmain, M., Mihailowski, N., Prajapati, V., Shah, K., and Remy, N. (2014). *RGoogleAnalytics: R Wrapper for the Google Analytics API*. R package version 0.1.1.

Prem, E., Sanz, F. S., Lindorfer, M., Lampert, D., and Irran, J. (2016). Open Digital Science. Technical report, eutema GmbH (Austria) in co-operation with ZSI and Universidad de Zaragoza. available at https://www.researchgate.net/publication/303855957.

Rijsbergen, C. J. V. (1979). *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition.

Robertson, S. (2004). Understanding inverse document frequency: On theoretical arguments for IDF. *Journal of Documentation*, 60(5):503–520.

Rousseeuw, P. J. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65.

Ryan, J. A. (2016). *quantmod: Quantitative Financial Modelling Framework*. R package version 0.4-7.

Salton, G. (1968). *Automatic Information Organization and Retrieval*. McGraw Hill Text.

Salton, G., Wong, A., and Yang, C. S. (1975). A Vector Space Model for Automatic Indexing. *Communications of the ACM*, 18(11):613–620.

Sanderson, M. (2010). Test Collection Based Evaluation of Information Retrieval Systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375.

Scheidegger, C. (2016). *github: github API*. R package version 0.9.8.

Scheidegger, C. and Borke, L. (2017). *rgithubQ: GitHub API bindings for R - QuantNet edition*. R package version 0.5.0.

Srivastava, A. and Sahami, M. (2009). *Text Mining: Classification, Clustering, and Applications*. Chapman & Hall/CRC, 1st edition.

Steinbach, M., Karypis, G., and Kumar, V. (2000). A comparison of document clustering techniques. In *In KDD Workshop on Text Mining*.

Stodden, V. C., Leisch, F., and Peng, R. D. (2014). *Implementing Reproducible Research*. CRC Press, London.

Theußl, S., Feinerer, I., and Hornik, K. (2012). A tm Plug-In for Distributed Text Mining in R. *Journal of Statistical Software*, 51(5):1–31.

Tibshirani, R. (1996). Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Tierney, L., Rossini, A. J., Li, N., and Sevcikova, H. (2016). *snow: Simple Network of Workstations*. R package version 0.4-2.

Warnes, G. R., Bolker, B., Bonebakker, L., Gentleman, R., Liaw, W. H. A., Lumley, T., Maechler, M., Magnusson, A., Moeller, S., Schwartz, M., and Venables, B. (2016). *gplots: Various R Programming Tools for Plotting Data*. R package version 3.0.1.

Weiss, S. M., Indurkhya, N., and Zhang, T. (2010). *Fundamentals of Predictive Text Mining*. Springer London.

Wickham, H. (2009). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

Wickham, H. (2015). *R Packages*. O'Reilly Media, Inc., 1st edition.

Widgren, S. and others (2016). *git2r: Provides Access to Git Repositories*. R package version 0.14.0.

Wild, F. (2015). *lsa: Latent Semantic Analysis*. R package version 0.73.1.

Wild, F. and Stahl, C. (2007). Investigating Unstructured Texts with Latent Semantic Analysis. In Decker, R. and Lenz, H. J., editors, *Advances in Data Analysis. Proceedings of the 30th Annual Conference of the Gesellschaft für Klassifikation e.V., Freie Universität Berlin, March 8-10, 2006*, pages 383–390, Berlin Heidelberg. Springer.

Witten, I. H., Moffat, A., and Bell, T. C. (1999). *Managing Gigabytes: Compressing and Indexing Documents and Images, Second Edition*. Morgan Kaufmann.

Wong, S. K. M., Ziarko, W., and Wong, P. C. N. (1985). Generalized Vector Spaces Model in Information Retrieval. In *Proceedings of the 8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '85, pages 18–25, NY, USA. ACM.

Xie, Y. (2016a). *formatR: Format R Code Automatically*. R package version 1.4.

Xie, Y. (2016b). *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.15.

Yi, C. (2016). *hqreg: Regularization Paths for Lasso or Elastic-Net Penalized Huber Loss Regression and Quantile Regression*. R package version 1.3.

Yi, C. and Huang, J. (2015). Semismooth Newton Coordinate Descent Algorithm for Elastic-Net Penalized Huber Loss Regression and Quantile Regression. *ArXiv e-prints*. 1509.02957.

Yu, L., Härdle, W. K., Borke, L., and Benschop, T. (2017). FRM: a Financial Risk Meter based on penalizing tail events occurrence. *SFB 649 Discussion Paper*. Humboldt Universität zu Berlin.

Selbständigkeitserklärung

Ich bezeuge durch meine Unterschrift, dass meine Angaben über die bei der Abfassung meiner Dissertation benutzten Hilfsmittel, über die mir zuteil gewordene Hilfe sowie über frühere Begutachtungen meiner Dissertation in jeder Hinsicht der Wahrheit entsprechen.

Berlin, den 7. September 2017 Lukas Borke